2021

Establishing behavioral baselines for computational systems: two case studies

John Henry Ring
University of Vermont

Establishing behavioral baselines for computational systems: two case studies

A Dissertation Presented

by

John H. Ring IV

to

The Faculty of the Graduate College

of

The University of Vermont

In Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy
Specializing in Computer Science

May, 2021

Defense Date: February 24th, 2021
Dissertation Examination Committee:

Christian Skalka, Ph.D., Advisor
Brian Tivnan, Ph.D., Chairperson
Joseph Near, Ph.D.
Safwan Wshah, Ph.D.
Cynthia J. Forehand, Ph.D., Dean of Graduate College

Abstract

The behavior of modern systems lives in a complex landscape that is unique to its particular application. In this work we describe and analyze the behavior of two modern computational systems: a Linux server and the National Market System (NMS). Though this work is diverse in both the type and scale of system under study, it is unified through the design and implementation of computationally tractable quantitative metrics aimed at defining the state of behavior of these systems. Understanding the behavior of these systems allows us to ensure their desired operation. In the case of a server we need to quickly be alerted when the system is compromised. Similarly, we need to know when a systematic or structural change in the NMS has unintended side-effects.

We first explore methods for host-based intrusion detection. *Host-based Intrusion Detection Systems* (HIDS) automatically detect events that indicate compromise of the host by adversarial applications. We propose and implement a full pipeline for HIDS development on an arbitrary host system. Our methodology first learns the sequence structure in system calls on an uncompromised host by predicting future calls. We then use predictions from this model to detect anomalies at the application level. Our pipeline is evaluated on an existing event sequence corpora, and PLAID. The **P**LAID **L**ab **A**rtificial **I**ntrusion **D**ataset is a new corpus for HIDS development we created to be more representative of modern systems. In addition, we characterize differences in attack and baseline behavior using allotaxonographs.

Next we turn our attention to the NMS for which we propose measures to quantify inefficiencies resulting from the geographic fragmentation of the equity marketplace. Using the most comprehensive, commercially-available dataset of trading activity in U.S. equity markets, we catalog and analyze quote dislocations between the SIP National Best Bid and Offer (NBBO) and a synthetic BBO constructed from direct feeds. We observe a total of over 3.1 billion dislocation segments in the Russell 3000 during trading in 2016, roughly 525 per second of trading. These dislocations do not behave as expected, often persisting meaningfully longer and with higher magnitude than what physical constraints suggest. These dislocations exhibit a characteristic structure that features more dislocations near the open and close. Around 23% of observed trades executed during dislocations leading to estimated opportunity costs on the order of $2 billion USD. A subset of the constituents of the S&P 500 index experience the greatest amount of opportunity cost and appear to drive inefficiencies in other stocks.

Acknowledgements

First I would like to thank my dissertation committee: Joe Near, Chris Skalka, Brian Tivnan, and Safwan Wshah who have proven themselves to be excellent educators, mentors, and colleagues.

Other members of the UVM community—Chris Danforth, Peter Dodds, Laurent Hebert-Dufresne, and Jean-Gabriel Young; who have greatly influenced my work and expanded my horizons. Fellow members of PLAID, past and present, including Sam Clark, Sam Durst, Tim Stevens, Krystal Maughan. My former colleagues at MITRE including: Matt Koehler, Matt McMahon, David Slater, and Jason Veneman, whoes efforts made much of this work possible.

Those who traveled the entire UVM journey by my side, David Dewhurst and Colin Van Oort; as well as colleagues who became friends along the way—Thayer Alshaabi, Sophie Gonet, Kelly Gothard, Ari Larson, Josh Minot, Vanessa Myhaver, and Anne Marie Stupinski.

My immediate family—Ilianna Ring, John Ring III, Ann Demopoulos; grandparents—John Ring Jr., Nora Ring, June Demopoulos; and the rest of my extended family, who have always supported me. And Finally, I thank Amanda Kurtz, for providing love and support while I ramble about data.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Researchers have long studied the behavior of complex computational systems. In the cyber-security realm *Behavioral Based Security* (BBS) approaches are used to detect intrusions or misuse. While in financial markets participants such as traders, exchanges, and regulators aim to discover *Stylized Facts*, empirical findings that describe the system's behavior. These market participants then apply their behavioral understanding toward a wide variety of ends ranging from profit generation to mechanism design.

A common cyber-security application of BBS by both academia [5] and industry [6, 7, 8] is the development of *Intrusion Detection Systems* (IDSs); tools that automatically detect events indicating system compromise by malicious adversaries. IDSs fall into one of two main categories based on their detection methodology: *signature-based* or *anomaly-based*. The key difference in these two approaches is what behavior is being observed and how that information is used. Signature based approaches operate similarly to a virus scanner: they report events matching a signature, that is a pattern of behavior, of a known attack. For example, the MITRE ATT&CK

Framework [9] is a set of signatures, expressed as rules for detecting intrusions, that can be used to flag events for further examination. Anomaly based approaches model normal system behavior and report *abnormal* events. Signature-based IDS offer a low false alarm rate, and do not require modeling of the system, but are unable to detect novel attacks. The ability to detect novel attacks—i.e. ones that have not been previously encountered—is the key advantage of anomaly-based systems.

IDSs are additionally classified by the data source they analyze. Host-based IDS (HIDS) monitor local events on its own internals and interfaces. Network-based Intrusion Detection Systems (NIDS) examine *network* events (i.e. traffic between hosts), rather than events occurring on a single host, and are thus distinct from HIDS. NIDS have traditionally been simpler to deploy than HIDS, since they do not require modifying individual hosts. However, as important services increasingly migrate to the cloud—where the network is under the control of the cloud provider—deploying a network-based approach for intrusion detection is often not feasible. The relative importance of HIDS research in the intrusion detection space is therefore increasing with the use of cloud computing.

The behavior of financial markets and their participants has long been of interest to academics, traders, exchanges and regulators [10]. Securities markets, such as the NMS, utilize auction mechanisms to facilitate the valuation and trade of assets [11, 12, 13, 14]. The NMS, as of 2016, was comprised of 13 networked exchanges coupled by information feeds of differential quality and subordinated to national regulation. Adding another layer of complexity, the NMS supports a diverse ecosystem of market participants, ranging from small retail investors to institutional financial firms and designated market makers. Market participants are classified primarily by

their trading behavior [15, 16, 17].

Efficiency is perhaps the most studied behavioral trait of modern markets. To "maintain fair, orderly, and efficient markets ..." [18] is a primary goal of market's chief regulatory body, the Securities Exchange Commission (SEC), and of exchanges who are competing against each other for market-share. Implementation details of these markets, including the auction mechanism, computing and communication infrastructure, as well as information dissemination policies, impact their informational and economic efficiency [19, 20]. The impact of market microstructure factors on high-level outcomes has been increasingly considered in recent analyses of market efficiency [21, 22, 23, 24]. This increased attention to market microstructure is due in part to the rise of High-Frequency Traders (HFT) who are categorized by their ultra fast timescales and sophisticated strategies that are dependent on the underlying microstructure.

A main goal of this dissertation is to demonstrate means to understand the behavior of real, large-scale computational systems. To that end we study both a modern server deployment and the NMS. Logging the events these systems perform, be it systems calls executed or financial orders placed, results in massive datasets cataloging the operation of these systems. Using modern data science tools such as machine learning and big data analytics with these datasets we can categorize and quantify the behavior of the underlying system.

The behavior we wish to quantify of course depends on our end goals, such as determining how a new regulation impacts market efficacy. To quantify this behavior we need to collect a dataset upon which to run our analysis. Finally, we need a measure of success to evaluate both our analysis and the underlying metric. This

is a highly interconnected and creative process, the data we collect determines what analysis can be performed and vice versa. We discuss our goals for analyzing these two systems in Sections 1.1 and 1.2. Throughout this dissertation we discuss the decisions that were made to achieve these goals along with the rationale of our chosen approaches.

1.1 Application 1: Host-Based Intrusion Detection with Deep Learning

We improve the state of the art for *Host-based Intrusion Detection Systems* (HIDS) utilizing *anomaly-detection* [25]. *Intrusion Detection Systems* (IDS) aim to automatically detect events indicating system compromise by malicious adversaries. Due to the growing importance of security threats, this problem has received considerable attention both in academic research [5] and from industry [6, 7, 8]. HIDS are a class of Intrusion Detection Systems (IDS) that monitor a computer system's internals and interfaces to detect intrusions. Systems that utilize anomaly-detection model normal system behavior and report abnormal events. The primary alternative to anomaly-based IDSs is *signature-based*. Signature based approaches operate similarly to a virus scanner: they report events matching the signature of a known attack. For example, the MITRE ATT&CK Framework [9] is a set of signatures, expressed as rules for detecting intrusions, that can be used to flag events for further examination. Unlike signature-based approaches, anomaly-based approaches can detect novel attacks, as they are identifying changes in behavior rather than a specific attack.

Network-based Intrusion Detection Systems (NIDS), the primarily alternative to

HIDS, examine *network* events (i.e. traffic between hosts), rather than events occurring on a single host, and are thus distinct from HIDS. NIDS have traditionally been simpler to deploy than HIDS, since they do not require modifying individual hosts. However, as important services increasingly migrate to the cloud—where the network is under the control of the cloud provider—deploying a network-based approach for intrusion detection is often not feasible. The relative importance of HIDS research in the intrusion detection space is therefore increasing with the use of cloud computing. We chose to focus on anomaly-based HIDS to create systems compatible with modern cloud deployments that can protect against zero-day attacks.

Automated methods for HIDS are generally formulated as analyses of sequences of system events such as bash commands or system calls [5]. System calls are the interface for userspace programs to request services from the operating system's kernel, such as starting a new process or reading a file. In HIDS research, system call sequences are used as a proxy for understanding the behavior of a running program—we assume that a malicious program will produce a very different pattern of system calls than baseline execution of a benign program. We focus on the use of machine learning to distinguish between malicious and baseline behavior in sequences of system calls.

1.2 Application 2: Inefficiencies in the National Market System

In our second application we explore how the behavior of the NMS is impacted by geographic fragmentation. We quantify the behavior for both individual securities and mutually exclusive groups, highlighting significant differences for securities trading on

Category	Duration	Magnitude	Count
	-	-	120,355,462
Dow	$> 545\mu s$	-	65,073,196
	$> 545\mu s$	$> 1¢$	2,872,734
	-	-	1,126,186,592
SPexDow	$> 545\mu s$	-	530,499,458
	$> 545\mu s$	$> 1¢$	51,187,430
	-	-	1,888,686,248
RexSP	$> 545\mu s$	-	704,416,718
	$> 545\mu s$	$> 1¢$	110,447,787

Table 1.1: Total number of dislocation segments in mutually-exclusive market categories. Number of opportunities is calculated unconditioned, conditioned on duration, and conditioned on both duration and magnitude.

identical microstructure [26, 27]. We investigate a broad subset of the equities traded in the U.S. National Market System (NMS), a network of stock exchanges located in the U.S., since it is the proverbial center of the world equity markets. In particular, we focus on constituents of the Russell 3000 Index, which is compiled by FTSE International Ltd. and contains roughly 3000 of the largest equities traded on the NMS. The selected sample represents the vast majority of the equities traded in the U.S. and can serve as a nearly comprehensive cross-section of publicly traded equities from which the observation and assessment of microstructure behaviors can be made.

We take a first-principles approach by compiling an exhaustive catalog of every dislocation, defined as a nonzero pairwise difference between the prices displayed by the National Best Bid and Offer (NBBO), as observed via the Securities Information Processor (SIP) feed, and Direct Best Bid and Offer (DBBO), as observed via the consolidation of all direct feeds. The SIP and consolidation of all direct feeds are representative of the displayed quotes from the national exchanges (lit market).

1	Realized Opportunity Cost	$2,051,916,739.66
2	SIP Opportunity Cost	$1,914,018,654.41
3	Direct Opportunity Cost	$137,898,085.25
4	Trades	4,745,033,119
5	Diff. Trades	1,124,814,017
6	Traded Value	$28,031,002,997,692.75
7	Diff. Traded Value	$7,077,357,462,641.67
8	Percent Diff. Trades	23.71
9	Percent Diff. Traded Value	25.25
10	Ratio of 9 / 8	1.0651

Table 1.2: Summary statistics of the realized opportunity cost (ROC) aggregated across all studied securities and all of calendar year 2016. The total ROC of this sample is over $2B USD. We discuss statistical characteristics of ROC extensively in Section 6.2. Row 10 shows that the average differing trade moves approximately 6.51% more value than the average trade. This indicates a qualitative shift in trading behavior during dislocations.

Additionally, we catalog every trade that occurred in the NMS among our selected sample in calendar year 2016, allowing an investigation of the relationship between trade execution and dislocations. We compile a dataset of all trades that may lead to a non-zero realized opportunity cost (ROC). We find that dislocations—times during which best bids and offers (BBO) reported on different information feeds observed at the same time from the point of view of a unified observer differ—and differing trades—trades that occur during dislocations—occur frequently. We measure over 3 billion dislocation segments (DSs), events derived from dislocations between the NBBO and DBBO. Table 1.1 shows that approximately 1.3 billion of those dislocation segments are what we term *actionable*, meaning that we estimate that there exists a nontrivial likelihood that an appropriately equipped market participant could realize arbitrage profits due to the existence of such a dislocation segment. We term any trade that executes during a dislocation a differing trade. Row 8 of Table 1.2 shows that 23.71% of all trades were differing trades. Differing trades may have been

influenced by stale quote information, so we used them to calculate realized opportunity costs (ROC). However, some trades may have been executed in this period intentionally, so we only include differing trades that executed at either of the two NBBO quotes. This results in a conservative estimate of total ROC, $2,051,916,739.66 across the Russell 3000 in 2016, as depicted in Row 1 of Table 1.2.

To facilitate this analysis we use the most comprehensive dataset of NMS messages commercially available which is effectively identical to that used by the Securities and Exchange Commission's (SEC) Market Information Data Analytics System (MIDAS). In addition to its comprehensive nature, this data was collected from the viewpoint of a unified observer: a single and fixed frame of reference co-located from within the Nasdaq data center in Carteret, N.J. We are unaware of any other source of public information (i.e., dataset available for purchase) or private information (e.g., available only to government agencies) that is collected using the viewpoint of a single, unified observer. Additionally, we note that despite recent technological upgrades to market infrastructure, the chief economist at Nasdaq confirms that our bar for actionability remains material in 2020 for the execution of latency arbitrage strategies [28].

We demonstrate that the topological configuration of the NMS entails endogenous inefficiency. The fractured nature of the auction mechanism, continuous double auction operating on 13 heterogeneous exchanges and at least 35 Alternative Trading Systems (ATSs) [29], is a consistent generator of dislocations and opportunity cost realized by market participants. The quantification of these dislocations establishes a baseline to benchmark the effect of new trading venues and regulatory changes have on market efficacy. This is especially relevant since as of writing in late 2020 two new venues, the Members Exchange, and the Long Term Stock Exchange are coming

online.

1.3 Contributions

To summarize, our primary contributions span two distinct, though related applications. In the first application we propose an alternative to trace-level anomaly detection with the *ALAD* algorithm, manufacture a new dataset, and apply advanced data visualization techniques. In the second application we provide concrete definitions for DS and ROC, apply and compare these measures across the constituents of collated equity indices, and propose novel visualization techniques. The similarity of the contributions for these two applications highlights the parallels between these domains. In both cases we collect sequences of events, apply a metric on these sequences, then compare the results by sequence classification.

Chapter 2

Literature Review

2.1 Intrusion Detection Systems

Intrusion detection systems (IDS) aim to automatically detect events indicating system compromise by malicious adversaries and have been studied since at least 1980 [30]. Liu and Lang provide a comprehensive taxonomy of the systems developed since then [5]. IDS are typically classified according to their *sources of data* and *detection methods*.

Network- vs. host-based intrusion detection. There are two major categories of data sources. Network-based intrusion detection systems (NIDS) are deployed at the network level, and detect intrusions by examining network traffic. Host-based intrusion detection systems (HIDS), which are the subject of this work, are deployed on a single host and detect intrusions by examining events on that individual host. NIDS have traditionally received more attention (e.g. [31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45]) because they are easier to deploy, more efficient, and capable of

detecting threats across multiple hosts. HIDS have the advantage of being deployable in a cloud setting, in which the cloud provider controls the network infrastructure, and are capable of detecting intrusions that do not produce abnormal network traffic. Our work focuses on HIDS.

Data & datasets. Our work is focused on detecting intrusions using sequences of system calls. System calls are the interface for userspace programs to request services from the operating system's kernel, such as starting a new process or reading a file. Forrest et al. first proposed using these sequences to detect intrusions, by collecting information about "normal" patterns of system calls and detecting system call sequences that deviate from these patterns [46]. Datasets of system call sequences include both *baseline* and *attack* sequences. Baseline sequences are collected from programs running normally; attack sequences are collected from compromised programs behaving abnormally (e.g. while an exploit is being used to attack the program).

Datasets of system call sequences are difficult to construct; as a result, most work in this area is evaluated on just four datasets:

- The DARPA Intrusion Detection Dataset [47] (1998/1999)
- The KDD 99 Dataset [48] (1999)
- The UNM System Call Dataset [49] (1998)
- The ADFA-LD Dataset [50] (2012)

Unfortunately, the DARPA, KDD, and UNM datasets are too old to be of practical use as representative of modern host processes and attacks [51]. The ADFA-LD (Australian Defence Force Academy Linux Dataset [50]) dataset was specifically designed

to address limitations of previously-collected datasets. In particular, they captured system call traces on a server running a modern operating system (Linux) with realistic workloads (e.g. web browsing and word processing), and attack sequences generated via real vulnerabilities in commonly-used software. For these reasons, the ADFA-LD dataset is often used for HIDS research, and previous work has demonstrated that this realism translates into a much more challenging learning task, suggesting that realistic datasets are vital for designing systems for practical deployment.

Nonetheless, the ADFA-LD dataset has a number of shortcomings. Since its release in 2012, typical workloads on Linux servers have changed, so the dataset is no longer reflective of typical server behavior. The dataset was captured on an i386 host, which though common at the time are rare in modern production environments. This is important because the system calls used by i386 and x86_64 systems differ substantially which makes it difficult to directly compare or integrate ADFA-LD traces with those collected on modern systems. Finally, the normal traces appear to be more reflective of a workstation, rather than server environment and are underspecified. Each attack sequence is labeled with the process which generated it, but the baseline sequences are not similarly labeled—so it is impossible to know what program was used to generate each sequence.

Signature- vs. anomaly-based methods. As mentioned earlier, there are two major methods of detection in HIDS research: *signature-based* methods and *anomaly-based* methods. Signature-based methods are commonly used to detect malware [52, 53, 54]; though they may also be used to detect known patterns of behavior that indicate an intrusion [45, 55]. These methods typically have low false-positive rates and are efficient, but they can only detect known attacks. Anomaly-based methods

detect abnormal behavior by comparing against a model of normal behavior; they have higher false positive rates, but are capable of detecting brand-new attacks. Anomaly-based methods have been applied both to sequences of system calls and to other kinds of intrusion detection [56, 57, 58, 59, 60, 61]. Our work focuses on anomaly-based intrusion detection.

IDS based on machine learning. A number of machine learning-based intrusion detection systems have been proposed by other authors. Machine learning approaches based on *supervised learning* (e.g. [45, 55, 62]) correspond to signature-based intrusion detection: they use labeled training data including both baseline behavior and attacks to train classifiers that distinguish between the two. These approaches cannot detect new kinds of attacks. Approaches based on *unsupervised learning* (e.g. [63, 64, 65, 66, 67, 68, 69, 70]) correspond to anomaly-based intrusion detection: they train models of baseline behavior using unlabeled training data containing only baseline behavior. Our work focuses on the use of unsupervised deep learning to perform anomaly-based intrusion detection on system call sequences. Previous work in this area has used both traditional ("shallow") machine learning and deep learning to build models of benign system call sequences. For example, approaches based on Hidden Markov Models [63, 64, 65] and support vector machines (SVM) [66, 67] have both been proposed. These methods worked well on datasets collected in the 1990's but performed poorly on the more recent ADFA-LD [50]. In particular, methods that discard the ordering information in system call sequences, including clustering and "bag of system calls" approaches achieve reasonable accuracy on legacy datasets but fail on ADFA-LD. Due to this recent approaches focus on techniques that leverage ordering information, of which deep-learning has been shown to be the most promising. Kim et al. compared

a long short-term memory (LSTM) model which k-nearest neighbor and k-means clustering achieving state-of-the-art performance with the LSTM [68]. Chawla et al. use a combined convolutional / recurrent (CNN / RNN) architecture, and obtain similar performance LSTMs with less training time [70]. These deep-learning based approaches represent the state-of-the-art in anomaly-based HIDS, and we use them for comparison in our empirical evaluation.

Visualisation Various visualization techniques have been used to aid human analysts and users in identifying suspicious activities and emerging threats in the cyber-security realm [71, 72]. Recent work in the field of Complex Systems provides analytical methods and corresponding visualizations for comparing various states of a system [73]. These advances have not previously been applied in the cyber-security domain though the divergent nature of attack vs baseline system call sequences is a natural fit for the application.

2.2 Financial Markets

Empirical Studies of Modern U.S. Markets In a recent report to its government oversight committee, the U.S. Securities and Exchange Commission (SEC) offered the following characterization of the prevailing literature which relates to our study: "It is unsurprising that academic studies generally are narrowly focused, as the amount of data, computing power and sophistication necessary to engage in broader study are daunting and costly, and relevant data may not be widely available or easily accessible." [74]. Given these constraints, we are aware of only two other recent studies which also used comprehensive, market data to analyze modern U.S. market

behavior and develop stylized facts.

In the first study, Wah [75] calculated the potential opportunities for latency arbitrage on the S&P 500 in 2014 using data from the SEC's MIDAS platform [76]. Using similar data to that for our study, Wah identified price discrepancies that could serve as latency arbitrage opportunities. Wah located time intervals during which the highest buy price on one exchange was higher than the lowest sell price on another exchange, termed a "latency arbitrage opportunity" in that work, and examined the potential profit to be made by an infinitely-fast arbitrageur taking advantage of these price discrepancies. Wah estimates that this idealized arbitrageur could have captured $3.0B USD from latency arbitrage in 2014, which is similar to our conservative calculations of approximately $2.1B USD in ROC from actual trades in 2016.

The second study was Aquilina et al. [77], which used message data from 2015 to quantify aspects of latency arbitrage in global equity markets. The authors note the frequent yet fleeting occurrence of latency arbitrage opportunities and estimate profits from latency arbitrage in 2018 at $4.8B USD globally, including $2.8B USD in the U.S. equity market.

Both the Wah and Aquilina et al. studies relied on affiliations with regulatory agencies and their respective data. This reliance on regulatory data supports the SEC observation that "relevant data may not be widely available or easily accessible."

Theory of market efficiency The studies above suggest that markets are not perfectly efficient. This proposition is further supported by O'Hara [78], Bloomfeld [79], Budish [80], who provide evidence that well-informed traders are able to consistently beat market returns as a result of both structural advantages and the actions of

less-informed traders, so called "noise traders" [81]. The suggestion that an implementation of a computation system is not perfectly efficient comes to no surprise to computer scientists but appears, at least at face value, to contradict prevailing economic theory.

The efficient markets hypothesis (EMH) as proposed by Fama [82] states that asset prices reflect all available information - the typical definition of market efficiency.. Thus, under this hypothesis it is impossible to systematically outperform the market since prices should only adjust when new information is presented. This hypothesis comes from analysis of 1960's and early 1970's transaction data. A stronger version of the EMH proposes the incorporation of private information as well, via insider trading and other mechanisms.

Previous studies have identified exceptions to this hypothesis [83], such as price characteristics of equities in emerging markets [84], the existence of momentum in the trajectories of equity prices [85], and speculative asset bubbles. Recent work by Fama and French has demonstrated that the EMH remains largely valid [85] when price time series are examined at timescales of at least 20 minutes and over a sufficiently long period of time. However, the NMS operates at speeds far beyond that of human cognition [86] and consists of fragmented exchanges [78] that may display different prices to the market. More permissive theories on market efficiency, such as the Adaptive Markets Hypothesis [87], allow for the existence of phenomena such as dislocations due to reaction delays, faulty heuristics, and information asymmetry [19]. In line with this, the Grossman-Stiglitz paradox [88] claims that markets cannot be perfectly efficient in reality, since market participants would have no incentive to obtain additional information. If market participants do not have an incentive to ob-

tain additional information, then there is no mechanism by which market efficiency can improve. This compendium of results points to a synthesis of the competing viewpoints of market efficiency. Specifically, that financial markets do seem to eventually incorporate all publicly available information, but deviations can occur at fine timescales due to market fragmentation and information asymmetries.

Market Dislocations Fragmented markets, such as the NMS, cannot be perfectly efficient due to physical considerations alone. The speed of information propagation is bounded above by the speed of light in a vacuum making it impossible for information to propagate instantaneously to spatially separated matching engines. These physically-imposed information propagation delays lead us to expect some decoupling of BBOs across both matching engines and information feeds. Such divergences were found between quotes on NYSE and regional exchanges as long ago as the early 1990s [89], in NYSE securities writ large [90], in Dow 30 securities in particular [91], between NASDAQ broker-dealers and ATSs as recently as 2008 [92, 93], and in NASDAQ listed securities as recently as 2012 [22]. U.S. equities markets have changed substantially in the intervening years, hence the motivation for our research. It is *a priori* unclear to what extent dislocations should persist within the NMS beyond the round-trip time of communication via fiber-optic cable. A first-pass analysis of latencies between matching engines could conclude that, since information traveling at the theoretical speed of light between Mahwah and Secaucus would take approximately 372 μs to make a round trip between those locations, then dislocations of this length might be relatively common. However, a light-speed round trip between Secaucus and Mahwah takes approximately 230 μs and between Secaucus and Carteret takes approximately 174 μs. Enterprising agents at Secaucus could rectify the differences

in quotes between Mahwah and Carteret without direct interaction between agents in Carteret and agents in Mahwah.

Several other authors have considered the questions of calculating and quantifying the occurrence of dislocations or dislocation-like measures. In the aggregate, these studies conclude that price dislocations do not have a substantial effect on retail investors, as these investors tend to trade infrequently and in relatively small quantities, while conclusions differ on the effect of dislocations on investors who trade more frequently and/or in larger quantities, such as institutional investors and trading firms. Ding, Hanna, and Hendershot (DHH) [22] investigate dislocations between the SIP NBBO and a synthetic BBO created using direct feed data. Their study focuses on a smaller sample, 24 securities over 16 trading days, using data collected by an observer at Secaucus, rather than Carteret, and does not incorporate activity from the NYSE exchanges. They found that dislocations occur multiple times per second and tend to last between one and two milliseconds. In addition, DHH find that dislocations are associated with higher prices, volatility, and trading volume. A study by the TABB Group of trade execution quality on midpoint orders in ATSs also noted the existence of latency between the SIP and direct data feeds, as well as the existence of intra-direct feed latency, due to differences in exchange and ATS software and other technical capabilities [94].

High-Frequency Trading Other authors have analyzed the effect of high-frequency trading (HFT) on market microstructure, which is at least tangentially related to our current work due to its reliance on low-latency, granular timescale data and phenomena. O'Hara [78] provides a high-level overview of the modern-day equity market and in doing so outlines the possibility of dislocation segments arising from differential

information speed. Angel [95, 96] claims that price dislocations are relatively rare occurrences, while Carrion [97] provides evidence of high-frequency trading strategies' effectiveness in modern-day equity markets via successful, intra-day market timing. Budish [80] notes that high-frequency trading firms successfully perform statistical arbitrage (e.g., pairs trading) in the equities market, and ties this phenomenon to the continuous double auction mechanism that is omnipresent in the current market structure. Menkveld [98] analyzed the role of HFT in market making, finding that HFT market making activity correlates negatively with long-run price movements and providing some evidence that HFT market making activity is associated with increasingly energetic price fluctuations. Kirilenko [15] provided an important classification of active trading strategies on the Chicago Mercantile Exchange E-mini futures market, which can be useful in creating statistical or agent-based models of market phenomena. Mackintosh noted the effects of both fragmented markets and differential information on financial agents with varying motives, such as high-frequency traders and long-term investors, in a series of Knight Capital Group white papers [23]. These papers provide at least three additional insights relevant to this dissertation. The first is a comparison of SIP and direct-feed information, noting that "all data is stale" since, regardless of the source (i.e., SIP or direct feed), rates of data transmission are capped at the speed of light in a vacuum as discussed above. The second is that the SIP and the direct feeds are almost always synchronized. That is, for U.S. large cap stocks such as Dow 30 constituents, synchronization between the SIP and direct feeds existed for 99.99% of the typical trading day. Stated another way, Mackintosh observed dislocations between quotes reported on the SIP and direct feeds for 0.01% of the trading day, or a sum total of 23 seconds distributed throughout

the trading day. The third relevant insight from the Mackintosh papers reflects the significance of dislocations. Mackintosh observed that 30% of daily value typically traded during these dislocations.

For a more comprehensive review of the literature on high frequency trading and modern market microstructure more generally, we refer the reader to Goldstein et al. [99] or Chordia et al. [100]. Arnuk and Saluzzi [101] provide a monograph-level overview of the subject from the viewpoint of industry practitioners.

Trade Execution Our calculations provide a conservative estimate of ROC from actual trades in the U.S. equity markets in 2016. Therefore, we identify some relevant literature on trade execution [102]; namely, where and when trades occur. First, trading is not instantaneous. Delays, or latencies, exist throughout the NMS. Second, not all trading activity occurs at a national exchange or an ATS. Instead of routing an order to one of these market venues, a broker may execute the order against the broker's own inventory of that stock. This process of retaining customers' orders internal to the brokerage is called "internalization" [103]. In addition to matching customers' orders against the broker's inventory of a particular stock, internalization also includes instances when a broker may route customers' orders to a market-maker under a Payment for Order Flow (PFOF) agreement. Even without charging commissions for trades, brokers may generate revenue from executing trades via PFOF [104]. To mitigate potential conflicts of interest, each broker is required to ensure that its customers' orders execute against best prices, as determined by the NBBO.

Trade execution problems may still arise from PFOF. In a public statement announcing its fine against a prominent market-maker, the SEC noted the use of algorithms which were used to avoid paying best prices on internalized orders. Per

the SEC, "these algorithms were triggered when they identified differences in the best prices on market feeds, comparing the SIP feeds to the direct feeds from exchanges" [105]. The reader will note that this market state, what the SEC has identified as "differences in best prices on market feeds", is the very same state that we have defined here as a market dislocation.

PFOF remains a controversial practice. More recently, another market-maker settled allegations that it did not ensure best prices for the internalization of its customers' orders [106].

We found references to internalization and PFOF dating back to 1994, when annual revenues from PFOF exceeded $500M USD across all U.S. brokers [107]. Some studies identified the potential for conflicts of interest from PFOF, but claimed that these conflicts could be mitigated by the adoption of minimum tick sizes of a penny (i.e., decimalization) [108, 109]. Though the SEC adopted decimalization in 2000 [110], PFOF remains a lucrative practice. In the first half of 2020, four brokers in the U.S. generated more than $1B USD in revenue from PFOF [111].

Scaling Behavior As we examine the scaling behavior between DSs, market capitalization (MC), and ROC we discuss previous quantification's of scaling behavior in financial markets. Mandelbrot [112, 113] was one of the first to characterize the scaling properties of price returns in modern markets. The scaling of returns was later revisited by Stanley and Plerou [114], Cont [10], as well as Patzelt and Bouchard [115]. Beyond returns in price time series, additional financial variables have been found to display scaling properties. Market indices and foreign exchange rates [116] as well as share volume and number of trades [117] adhere to scaling properties.

Chapter 3

Background & Motivation

3.1 Host-Based Intrusion Detection

Previous work has developed HIDS that operate on individual traces of system call sequences [68, 69] using publicly available datasets [47, 48, 49, 50]. Some of these prior works are also based on anomaly detection [56, 57, 58, 59, 60, 61]. All of these works consider system call traces generated by an individual process; however, modern applications often use multiple processes, and modern attacks can impact one or more of these processes. Furthermore, existing system call corpora used to develop these HIDS are limited and outdated. Thus, the problems we address are how to modernize anomaly-based HIDS by incorporating analysis of multi-process applications, how to develop algorithms and evaluation methods more relevant to modern systems and attacks, and overall how to achieve more accurate detection of modern attacks.

We address these problems as follows. *First*, we present a novel approach for building HIDS based on unsupervised deep learning. State-of-the-art in this domain demonstrates that models based on Long Short Term Memory (LSTM) [68],

and Gated Recurrent Unit (GRU) [69] architectures outperform prior SVM-based approaches and hence are the most promising technology in this space. The key technical contribution of our approach is an application-level classifier, called *ALAD* (*A*pplication-*L*evel *A*nomaly *D*etection), to distinguish between baseline and malicious behavior. *ALAD* groups system call sequences by *program*—rather than by *process*, as was done in previous work [68, 69]. *ALAD* is simple to implement, and in our experiments produces a statistically significant improvement in classification compared to previous work. We describe the *ALAD* approach in Section 5.1.5.

Second, we collect and release a new dataset of system call sequences, with modern attacks on multi-process applications, used to support the development of our approach and validate our results. Our new dataset, called PLAID, contains sequences from six modern exploits and penetration techniques as well as a large collection from normal operation. We discuss the creation of PLAID in Section 4.1.

The *third* main contribution of our paper is the application and evaluation of modern sequence-to-sequence neural network architectures for anomaly detection. In Section 5.1, we compare a state-of-the-art architecture, WaveNet [118], with replications of the LSTMs and GRUs used in prior work, using both *ALAD* and the trace-level classifiers developed in previous work. We demonstrate our results on PLAID as well as the Australian Defence Force Academy Linux Dataset (ADFA-LD) [50], used by several closely related works [68, 69]. We completed 540 training and evaluation trials over combinations of dataset, model, and replicate. To our knowledge this is the largest comparison of deep learning models used in HIDS to date. We provide open source repositories for all datasets and code[1] to facilitate reproducibility.

[1]Repository link suppressed for double-blind review.

In addition, we address a common critique of deep learning, that it is "black-box", in the sense that it structurally obfuscates model details and does not provide practitioners with insights about *why* it works. We show in Section 5.4, that recent techniques in corpora "divergence" visualization can still provide useful insights into datasets. Specifically, we explore our new dataset along with the popular ADFA-LD to observe differences between normal and malicious sequences. This helps to explain the effectiveness of anomaly detection in this application.

In summary, our primary contributions are as follows:

1. Application Level Anomaly Detection (ALAD), a new classifier for *groups* of system call sequences.

2. PLAID, a new dataset of modern system call sequences and attacks.

3. A comparison of modern sequence-to-sequence neural network architectures for anomaly detection.

4. The use of rank-turbulence divergence to visualize differences in system-call n-grams.

Note that (3) also subsumes a comparison with historical work, since [68, 69] already demonstrated superiority of deep learning approaches as compared to other historical approaches.

3.2 The National Market System

To understand the behavior of the NMS one must first be familiar with the infrastructure components and some varieties of market participants. For this reason we

provide a brief overview of the NMS as it stood in 2016. In particular, we note the information asymmetry between participants informed by the Securities Information Processor and those informed by proprietary, direct information feeds.

The U.S. equities market, known as the National Market System, is composed of 13 National Securities Exchanges. Each exchange contributes to price discovery through the interactions of market participants, mediated by an auction mechanism. Another core component of the NMS is a collection of approximately 40 alternative trading systems (ATSs) [119], also known as dark pools. ATSs provide limited pre-trade transparency, which can allow market participants to reduce the market impact of their trades, but have limited participation in price discovery as a result. Each exchange and ATS accumulates orders whose execution conditions have not been met in an order book. Resting orders are matched with incoming marketable orders based on a priority mechanism, commonly price-time priority [120]. Traders often have access to a variety of order types that allow them to tailor how they interact with the market [121, 122, 123, 124]. The top of the book at each exchange, the resting bid with the highest price and the resting offer with the lowest price, is called the best bid and offer (BBO). BBOs from across the NMS are aggregated by one of the Securities Information Processors (SIP) to form the national best bid and offer (NBBO) [125, 126]. Under Regulation National Market System (Reg. NMS), trades must execute at a price that is no worse than the NBBO, though exceptions exist (e.g. intermarket sweep orders) [127].

Market participants in the NMS have several options of data products to fuel their trading decisions. In addition to the dissemination of the NBBO, each SIP provides data feeds containing all quotes and trades that occur in their managed securities.

Information feeds offered by each exchange, referred to as direct feeds, can provide similar information with lower latency than the SIP data feeds. Direct feeds can also provide additional data, such as the resting volume at all price points, commonly called depth-of-book information. Information asymmetries between data products lead to DSs, which can impact trading decisions and outcomes.

The NMS is regulated by the U.S. Securities and Exchange Commission (SEC), a federal agency, and self-regulated by the Financial Industry Regulatory Authority (FINRA), a professional organization. FINRA polices its members and ensures they adhere to SEC rules and other professional guidelines, while SEC designs, implements, and enforces rules that are intended to promote market stability and economic efficiency. The physical structure of the NMS, in conjunction with the existence and usage of multiple distinct information feeds, leads to the creation of DSs and associated ROC.

3.2.1 Market Participants

There are, broadly speaking, three classes of agents involved in the NMS: traders, of which there exist essentially four sub-classes (retail investors, institutional investors, brokers, and market-makers) that are not mutually exclusive; exchanges and ATSs, to which orders are routed and on which trades are executed; and regulators, which oversee trades and attempt to ensure that the behavior of other market participants abides by market regulation. We note that Kirilenko *et al.* claim the existence of six classes of traders based on technical attributes of their trading activity [15]. This classification was derived from activity in the S&P 500 (E-mini) futures market, not the equities market, but is an established classification of trading activity. It is not

possible to perform a similar study in the NMS since agent attribution is not publicly available. However, the Consolidated Audit Trail (CAT) is an SEC initiative (SEC Rule 613) that may provide such attribution in the future [128]. At the time of writing this framework was not yet constructed.

3.2.2 Physical Considerations

Contrary to its moniker, "Wall Street" is actually centered around northern New Jersey. The matching engines for the three NYSE exchanges are located in Mahwah, NJ, while the matching engines for the three NASDAQ exchanges are located in Carteret, NJ. The other major exchange families base their matching engines at the Equinix data center, located in Secaucus, NJ, except for IEX, which is based close to Secaucus in Weehawken, NJ. The location of individual ATSs is generally not public information. However, since there is a great incentive for ATSs to be located close to data centers (see sections 2.2 and 5.2), it is likely that many ATSs are located in or near the data centers that house the NMS exchanges. For example, Goldman Sachs's Sigma X^2 ATS has its matching engine located at the Equinix data center in Secaucus, NJ [129].

Since matching engines perform the work of matching buyers with sellers in the NMS, we hereafter refer to the locations of the exchanges by the geographic location of their matching engine. For example, IEX has its point of presence in Secaucus, but its matching engine is based in Weehawken; we locate IEX at Weehawken.

This geographic decentralization has a profound effect on the operation of the NM. We calculate minimum propagation delays between exchanges and are displayed in Table 3.1. In constructing Table 3.1 we use estimates of propagation delays in fiber

NMS Propagation Delay Estimates

	Carteret Mahwah	Mahwah Secaucus	Carteret Secaucus	Secaucus Weehawken
Straight-line Distance	34.55mi 55.60km	21.31mi 34.30km	16.22mi 26.10km	2.56mi 4.12km
Light speed, one-way	185.75μs	114.57μs	87.20μs	13.76μs
Light speed, two-way	371.50μs	229.14μs	174.40μs	27.52μs
Fiber, one-way	272.44μs	168.07μs	127.89μs	20.19μs
Fiber, two-way	544.88μs	336.14μs	255.78μs	40.38μs
Hybrid laser, one-way	-	-	94.50μs	-
Hybrid laser, two-way	-	-	189.00μs	-

Table 3.1: The speed of light is approximated by $186,000$ *mi/s (or* $300,000$ *km/s) and fiber propagation delays are assumed to be 4.9μs/km. These propagation delays form the basis for estimates of the duration required for a dislocation segment to be considered actionable, though these figures do not account for any computing delays and thus are lower bounds for the definition of actionable.*

optic cables provided by M2 Optics [130] as well as data center locations, distances between data centers, and one-way hybrid laser propagation delays from Anova Technologies [131].

In reality, the time for a message to travel between exchanges will be strictly greater than these lower bounds, since light is slowed by transit through a fiber optic cable, and further slowed by any curvature in the cable itself. The two-way estimates in Table 3.1 give a lower bound on the minimum duration required for a dislocation segment to be "actionable" and a more realistic estimate derived by assuming propagation through a fiber optic cable with a refractive index of 1.47 [130]. These estimates do not account for computing delays, which may occur at either end of the communication lines, in order to avoid speculation. In practice such computing delays will also have a material effect on which dislocation segments are truly actionable and will depend heavily on the performance of available computing

hardware.

Connecting the exchanges are two basic types of data feeds: SIP feeds, containing quotes, trades, limit-up / limit-down (LULD) messages, and other administrative messages complied by the SIP; and direct data feeds, which contain quotes, trades, order-flow messages (add, modify, etc), and other administrative messages. The direct data feeds operate on privately-funded and installed fiber optic cables that may have differential information transmission ability from the fiber optic cables on which the SIP data feeds are transmitted. Latency in propagation of information on the SIP is also introduced by SIP-specific topology (SIP information must travel from a matching engine to a SIP processing node before being propagated from that node to other matching engines) and computation occurring at the SIP processing node. Due to the observed differential latency between the direct data feeds and the SIP data feed and the heterogeneous distance between exchanges, dislocation segments are created solely by the macro-level organization of the market system. We note that in the intervening years since data was collected for analysis, the SIP has been upgraded substantially to lower latency arising from computation at SIP processing nodes. Our understanding of the physical layout of the NMS is depicted in Fig. 3.1 at a relatively high level.

There are three basic types of information flow within the NMS:

a. Direct feed information, which flows to anyone who subscribes to it. Direct feed information is associated with non-trivial costs (on the order of $\$130,000$ USD per month and so is used primarily by exchanges, large financial firms, and ATSs. Direct feed information thus flows to and from the exchanges (and the major exchange participants). We hypothesize that direct feed information

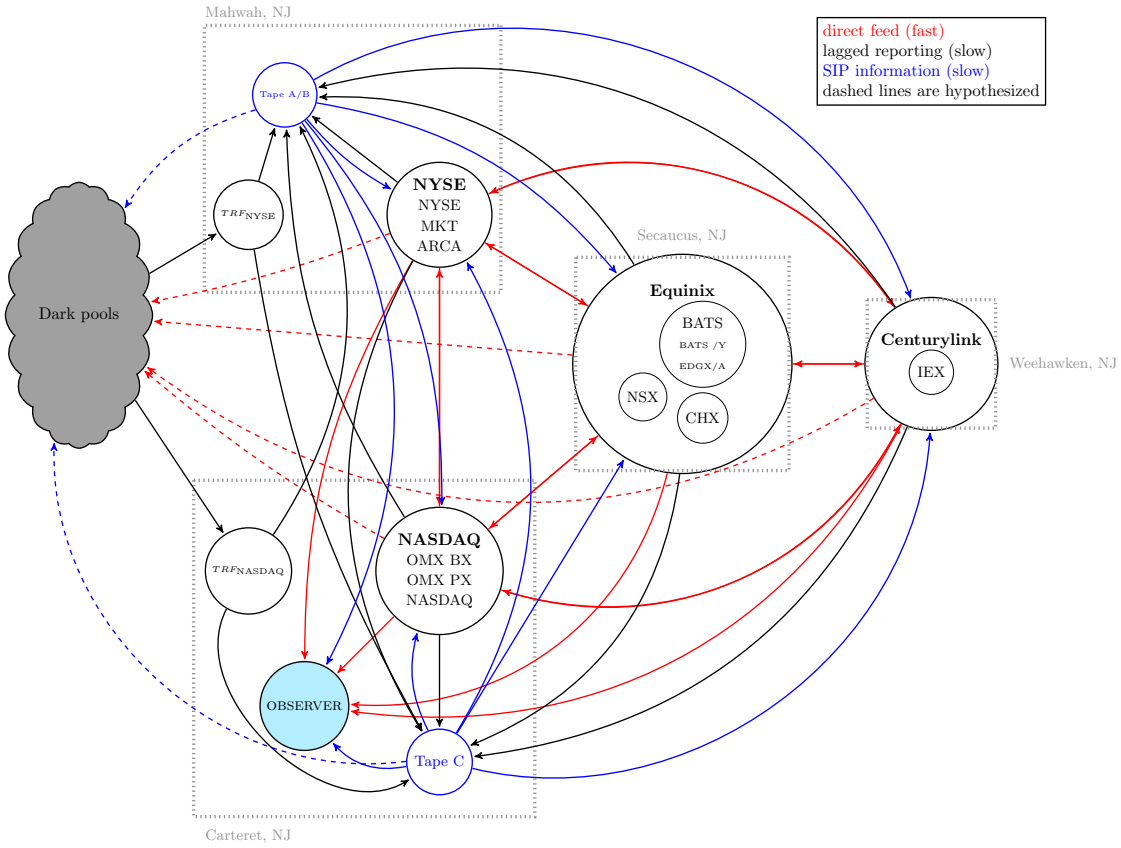


Figure 3.1: The NMS (lit market and ATSs) as implied by the comprehensive market data. As we do not have the specifications of inter-market center communication mechanisms and have minimal knowledge of intra-market center communication mechanisms, we simply classify information as having high latency, as the SIP and lagged information heading to the SIP do, or low latency, as the information on the direct feeds does. Note the existence of the observer, located in Carteret NJ. Without a single, fixed observer it is difficult to clock synchronization issues and introduces an unknown amount of noise into measurements of dislocations and similar phenomena. Clock synchronization issues are avoided when using data collected from a single point of presence since all messages may be timestamped by a single clock, controlled by the observer.

also flows to ATSs, since they require some type of price signal in order for the market mechanism to function and may benefit from low latency data. This was the case for at least one major ATS, Goldman Sachs's Sigma X^2, as of May 2019, so it is plausible that it is true for others [129]. The direct feeds provide the fastest means by which to acquire a price signal, and thus may provide the best economic value to traders dependent on frequent information updates; this provides the economic foundation for our hypothesis.

b. SIP information, which is considerably less expensive than direct feed information and exists by regulatory mandate. However, market participants may still subscribe to the SIP as a tool for use in arbitrage; see Section 2.2 for discussion of this possibility. Market participants that choose not to purchase the direct feed data might also choose to purchase the SIP data for use as a price signal and as a backup to the consolidated direct feeds. At least one ATS, Goldman Sachs's Sigma X^2, uses SIP data as a backup to direct feed data and combines both data sources to construct their local BBO [129].

c. Lagged reporting data that is not yet collated by the SIP. Regulation requires that exchanges report all local quote and trade activity, and that ATSs report all trade activity. This information is collected by the appropriate SIP tapes and then disseminated through the SIP data feeds. It is the responsibility of the exchanges to report their quote and trade information to the SIP, and of ATSs to report their trade information to FINRA Trade Reporting Facilities (TRF). Thus, though this information will be eventually visible to all subscribers to SIP or direct feed data, it differs qualitatively from that data due to its lagged nature.

For example, suppose a trade occurs at NYSE MKT on a NASDAQ-listed security that updates the NBBO for that security. Since this trade occurs at Mahwah, it takes a non-negligible amount of time for the information to propagate to SIP Tape C, located in Carteret. However, traders located at Mahwah have access to this information more quickly, possibly allowing them an information advantage over their Carteret-based competitors.

Chapter 4

Data

The collection and curation of datasets is perhaps the most important aspect of any application of data science. Knowledge of a systems behavior is learned through study of a dataset since said behavior is not defined through a discrete set of rules. Thus, one's findings are only as good as the underlying data, as the saying goes *garbage in, garbage out*.

In this dissertation we use multiple datasets. The core datasets catalog the systems operation, namely copra of system calls for our Linux server, and quote and trade messages for the NMS. Additionally, we use meta-information to isolate the behavior of aspects of the larger system. We discuss all datasets used and considered in our analysis here.

4.1 The PLAID Dataset

As with all machine learning techniques for IDS, our approach to training and testing models for HIDS relies on corpora of events, in our case system calls. Since we are

developing an anomaly detection system, training corpora must contain baseline and attack data as described above in Section 2.1. Given the shortcomings of ADFA-LD discussed in Section 2.1 we developed a new dataset, named PLAID, with modern system calls, and a richer, more current set of attacks. The **P**LAID **L**ab **A**rtificial **I**ntrusion **D**ataset is an open source dataset intended to support the work described here, and to support research in the broader community. PLAID features modern exploits carried out against a contemporary Linux server deployment, and is publicly available [132].

4.1.1 Host Configuration

Ubuntu 18.04 LTS [133] was selected as the host Operating System (OS) for PLAID. Ubuntu is a secure modern Linux distribution, and the most popular choice of OS for use on public clouds such as AWS and Microsoft Azure.

Commonly used remote administrative services FTP and SSH [134] were installed through Ubuntu's default package manager and enabled with their default configurations. Redis Version 4.0.14 [135] an open source in-memory data structure store was manually installed on the host and configured to allow connections on the local network. A malicious client side executable [136] was placed on the machine, simulating a successful social engineering attack. Nginx Version 1.14.0 [137] and php-fpm Version 7.1.33 [138] were installed on the host and configured to serve a basic website, a common deployment of the world's most popular web server [139].

This host configuration represents a reasonable approximation of a modern production Linux server offering remote access, high performance data storage, and web hosting.

4.1.2 Network Setup

Our experiment testbed consists of three Virtual Machines (VMs): our host, an attack machine, and a router. The attack VM is an instance of Kali 2019 [140], a Linux distribution designed for penetration testing. We connected our attack and host VMs on a local network through a bare-bones instance of Ubuntu 18.04 LTS serving as a router. All three VMs were run using VirtualBox Version 6.1 [141] on a single physical machine.

4.1.3 Attack Overview

Our host machine was exploited from six different attack vectors.

1. The **Redis attack** [142] exploits a vulnerability in the "extension" functionality provided in the Redis in-memory database to execute arbitrary code. An exploit for the vulnerability was developed in 2018 and is available in Metasploit.

2. The **PHP-FPM** attack [143] (CVE-2019-11043) exploits a vulnerability present in the combination of nginx and php-fpm to execute arbitrary code. An exploit for this vulnerability was developed in 2019 and is available on GitHub.

3. The **privilege escalation** attack [144] (CVE-2016-5195, also called DirtyCow) uses a malicious CSE that exploits a vulnerability in the Linux kernel to obtain a shell with root privileges.

4. The **brute-force** attacks [145] represent the use of a traditional brute-force password-cracking application (Hydra) to discover users' passwords over SSH and FTP.

4.1.4 Data Collection

System call traces were generated by starting the target application with `strace`—a userspace utility capable of monitoring interactions between processes and the Linux kernel. Each exploit was run and monitored for ten trials, fully restarting all affected services between each trial. The result of each trial is a series of files containing system calls for example: `execve brk access access openat fstat mmap close...` Each individual file corresponds to a single process of the program's execution and is labeled with the process id.

Since the intended use of this dataset is the development of anomaly-based IDSs, we require baseline data approximating normal operation. This baseline dataset was generated by monitoring a wide variety of common operations on the host with no active attacks in progress. Specific items in the set of common operations were chosen for two reasons. The first is to be representative of the wide range of computational tasks performed in modern day enterprise environments. The second is to achieve a high degree of behavioral overlap with the previously described attacks. The chosen baseline operations are:

- Transfer of files to and from the host using FTP
- Host access via SSH and modification of configuration files
- Simulation of web traffic using Apache Bench
- Redis interactions
- Download files from the internet with curl

- Execution of `rustup`, the Rust programming language install script [146]
- PHP and Redis test suites
- Compilation of small and large programs
- Deployment of small programs that involve: reading from disk, non-trivial computation, and standard IO

We encode meta-information in the directory structure in the same manner as the ADFAdataset. The generated data was split into two top level directories- attack and baseline. Inside the attack directory is a subdirectory for each trial labeled with the exploit and trial ID. These subdirectories contain all collected system call trace files from the corresponding exploit trial. Similarly, the baseline directory contains a subdirectory for each baseline operation. These subdirectories contain all collected system call traces associated with the baseline operation.

4.2 Equity Indices

Many of our results are centered around the components of three of the most popular equity indices: Dow Jones Industrial Average, S&P 500, and the Russell 3000. Indices measure the performance of a bucket of securities. The choice of the underlying securities is often to be representative of a market segment. Indices may not be directly purchased in the same way as an equity, but may be tracked by Exchange Traded Funds (ETFs) and mutual funds.

The Dow Jones Industrial Average, from here on referred to as the Dow, is a price weighted index that aims to provide an overview of the U.S. economy [147]. The Dow

consists of thirty S&P 500 constituents, covering all industries except for utilities and transportation.

The S&P 500 is a market capitalization weighted index of 500 large US based companies referred to by it's creators as "the gauge of the market economy". The index is considered by many to be representative of the US stock market as a whole and is a primary holding among passive investors. To be included in the index, as of 2016, a company must meet the following criteria [148].

- Be a U.S. Company
- Have a market capitalization greater than $5.3 billion
- Be highly liquid
- Have a public float of at least 50% of outstanding shares
- Had positive earnings in the most recent quarter
- The sum of the last four consecutive quarterly earnings must be positive
- Be listed on a major exchange

Meeting these criteria does not guarantee inclusion, and failing to uphold these standards does not necessarily result in immediate expulsion from the index. S&P 500 constituents are chosen by S&P Global, and the index is updated regularly, though not on any fixed schedule.

The Russell indexes are passively constructed (no human in the loop) based on a transparent set of rules including [149]:

- Be a U.S. Company

- Be listed on a major exchange
- Have a share price $\geq$ \$1
- Have a market capitalization $\geq$ \$30M
- Have a public float $\geq 5\%$

The Russell 3000 consists of the largest 3000 firms by market capitalization meeting the above criteria, or the entire eligible set, whichever is smaller. The index undergoes an annual reconstruction in June and is augmented quarterly with the addition of Initial Public Offerings (IPO). This methodology results in the Russell 3000 being a strict superset of the S&P 500.

For our analysis we focus on constituents of these indices, rather the index itself. Thus, differing weighting methodologies used by these indices have no effect on our analysis. We also note that some companies have multiple common stocks, one for each share class, and that each index handles the inclusion of multiple share classes differently.

4.3 NMS Dataset

We use a dataset comprised of every quote and trade message that was disseminated on one of the SIP or direct feeds during the period of study. This dataset features comprehensive coverage of the stocks under study, is collected from a single location (Carteret, NJ), and is time stamped upon arrival, thus limiting clock synchronization issues. Thesys Technologies collected and curated this data [150], and also provided data for the SEC's MIDAS [76] at the time of collection. Prior to awarding Thesys

Group the MIDAS contract, the SEC conducted a sole source selection [151], thereby designating Thesys Group as the only current authoritative source for NMS data.

The fact that this dataset was collected by a single fixed location allows us to directly observe market dislocations ROC. This is unlike previous studies where similar phenomenon could only be estimated. With the arrival timestamp we observe information flow through the NMS in the same manner as a market participant located at the Carteret data center. Ideally, we would have data from four different unified observers—an observer located at each data center—so that we could compile the different states of the market that must exist depending on physical location of observation, but we do not believe that comprehensive consolidated data is available from the point of view of observers located anywhere but at Carteret, hence our selection of this location for observation.

The securities under study are categorized by meta properties including: index membership, Global Industry Classification Standard (GICS) sector classification, and Market Capitalization (MC). Data on these properties was gathered using a standard commercial Bloomberg Terminal.

The indices we consider are subject to frequent changes in membership. To simplify our analysis we consider the Dow 30 and S&P 500 as they stood on Jan. 1, 2016. For the Russell 3000 we consider the constituents as listed in the June 2016 construction, excluding those that were not publicly traded on Jan. 1, 2016. Constituents of the indices under study were curated to only include companies that survived as a publicly traded entity on a national exchange for the entire calendar year of 2016. Companies that were delisted for any reason (e.g. bankruptcy or buyout) were excluded, in addition to those who were acquired by an out-of-study firm. Mergers

between in-sample companies did not result in exclusion. Curating the stocks under study in this way allows us to avoid issues caused by IPOs and delistings.

Many companies in our dataset changed their ticker symbol over the course of the calendar year and thus appear as a different entity in the data. To study a company over a long time period it is necessary to know all tickers it traded under and when the ticker changes occurred. There is no consolidated public record of these ticker changes, so we tracked them via an extensive review of press releases. These ticker changes were then validated by observing changes in trading activity in the old and new ticker on the date of the change using the Thesys data archive.

This curation reduced the Russell 3000 from 3005 stocks to 2903, the S&P 500 from 500 stocks to 472, and did not impact the 30 members of the Dow. We denote the curated version of an existing index by appending a prime to the respective base index (e.g. Dow 30 $\rightarrow$ Dow 30$'$). We then construct two additional stock groups, RexSP and SPexDow, by taking the appropriate set difference, e.g. SPexDow = S&P 500$'$ - Dow 30$'$. Finally, all companies in our dataset were classified by their MC as it stood in the beginning of Q4 2016 using the classes defined in Table 4.1. Our dataset covers approximately 98% of all publicly traded U.S. equities by MC [152]. Tables 4.2 and 4.3 provide summary statistics and distribution of these equities across GICS sector, MC, and market category, for several indices.

Class	Statistic	Russ 3K′	RexSP	S&P 500′	SPexDow	Dow 30′
Nano	% by #	0.14	0.16	0.00	0.00	0.00
	% by MC	0.00	0.00	0.00	0.00	0.00
	Count	4	4	0	0	0
Micro	% by #	11.51	13.74	0.00	0.00	0.00
	% by MC	0.26	1.09	0.00	0.00	0.00
	Count	334	334	0	0	0
Small	% by #	42.89	51.13	0.42	0.45	0.00
	% by MC	4.37	18.50	0.01	0.02	0.00
	Count	1,245	1,243	2	2	0
Mid	% by #	30.35	32.21	20.76	22.17	0.00
	% by MC	15.11	53.19	3.37	4.72	0.00
	Count	881	783	98	98	0
Large	% by #	14.50	2.71	75.21	75.79	66.67
	% by MC	56.68	20.72	67.77	77.59	43.28
	Count	421	66	355	335	20
Mega	% by #	0.62	0.04	3.60	1.58	33.33
	% by MC	23.58	6.50	28.85	17.67	56.72
	Count	18	1	17	7	10

Table 4.1: Composition of indexes under study by market capitalization (MC) classification as of Q4 2016. The composition of various indexes is displayed by the percentage of index constituents that are a member of each given index (% by #) and by the weighting of those constituents (% by MC).

Sector	Statistic	Russ 3K′	RexSP	S&P 500′	SPexDow	Dow 30′
Consumer Discretionary	% by #	14.92	14.52	16.95	17.19	13.33
	% by MC	12.97	16.40	11.92	13.10	8.98
	Count	433	353	80	76	4
	($) MC Min	95,330,024	95,330,024	1,244,719,232	1,244,719,232	84,654,022,656
	($) MC Max	356,313,137,152	89,539,158,016	356,313,137,152	356,313,137,152	165,862,064,128
Consumer Staples	% by #	4.03	3.41	7.20	7.01	10
	% by MC	8.54	3.83	9.99	9.69	10.74
	Count	117	83	34	31	3
	($) MC Min	114,570,432	114,570,432	9,794,159,616	9,794,159,616	178,815,287,296
	($) MC Max	224,997,457,920	17,508,790,272	224,997,457,920	150,058,582,016	224,997,457,920
Energy	% by #	5.20	4.73	7.63	7.69	6.67
	% by MC	6.57	4.71	7.14	5.83	10.40
	Count	151	115	36	34	2
	($) MC Min	160,502,160	160,502,160	2,427,903,232	2,427,903,232	222,190,436,352
	($) MC Max	374,280,552,448	27,468,929,024	374,280,552,448	116,800,331,776	374,280,552,448
Financials	% by #	17.81	18.84	12.50	12.44	13.33
	% by MC	15.17	21.99	13.07	14.73	8.91
	Count	517	458	59	55	4
	($) MC Min	89,903,488	89,903,488	3,021,111,552	3,021,111,552	34,774,474,752
	($) MC Max	401,644,421,120	401,644,421,120	308,768,440,320	276,779,139,072	308,768,440,320
Health Care	% by #	15.23	15.84	12.08	11.99	13.33
	% by MC	12.49	9.12	13.53	13.19	14.38
	Count	442	385	57	53	4
	($) MC Min	21,050,850	21,050,850	1,478,593,408	1,478,593,408	152,328,667,136
	($) MC Max	313,432,473,600	18,889,377,792	313,432,473,600	108,768,911,360	313,432,473,600
Industrials	% by #	13.47	13.41	13.77	13.57	16.67
	% by MC	10.40	11.03	10.20	9.91	10.94
	Count	391	326	65	60	5
	($) MC Min	58,695,636	58,695,636	2,821,674,240	2,821,674,240	54,259,630,080
	($) MC Max	279,545,937,920	13,281,452,032	279,545,937,920	100,041,220,096	279,545,937,920
Information Technology	% by #	14.40	14.60	13.35	12.90	20
	% by MC	21.40	13.81	23.74	20.93	30.74
	Count	418	355	63	57	6
	($) MC Min	114,370,240	114,370,240	3,334,570,240	3,334,570,240	151,697,113,088
	($) MC Max	617,588,457,472	32,402,583,552	617,588,457,472	538,572,161,024	617,588,457,472
Materials	% by #	4.55	4.40	5.30	5.43	3.33
	% by MC	3.26	5.83	2.47	3.02	1.11
	Count	132	107	25	24	1
	($) MC Min	103,733,456	103,733,456	2,823,849,728	2,823,849,728	63,809,703,936
	($) MC Max	69,704,540,160	69,704,540,160	63,809,703,936	46,132,944,896	63,809,703,936
Real Estate	% by #	6.61	6.99	4.66	4.98	0.00
	% by MC	3.89	8.67	2.41	3.38	0.00
	Count	192	170	22	22	0
	($) MC Min	161,591,616	161,591,616	7,130,559,488	7,130,559,488	0.00
	($) MC Max	55,830,577,152	24,264,243,200	55,830,577,152	55,830,577,152	0.00
Telecommunication Services	% by #	1.03	1.03	1.06	0.90	3.33
	% by MC	2.40	1.82	2.57	2.09	3.79
	Count	30	25	5	4	1
	($) MC Min	285,299,072	285,299,072	3,964,831,488	3,964,831,488	217,610,731,520
	($) MC Max	261,176,721,408	47,389,126,656	261,176,721,408	261,176,721,408	217,610,731,520
Utilities	% by #	2.76	2.22	5.51	5.88	0.00
	% by MC	2.91	2.78	2.95	4.13	0.00
	Count	80	54	26	26	0
	($) MC Min	141,720,064	141,720,064	3,867,331,328	3,867,331,328	0.00
	($) MC Max	57,253,351,424	12,880,323,584	57,253,351,424	57,253,351,424	0.00

Table 4.2: Market Capitalization (MC) statistics of equities under study broken out by Global Industry Classification Standard (GICS) sector as of Q4 2016. The composition of various indexes is displayed by the percentage of index constituents that are a member of each given sector (% by #) and by the weighting of those constituents (% by MC). Additionally, the MC of the smallest and largest constituent for each index in each category is displayed.

	Russ 3K′	RexSP	S&P 500′	SPexDow	Dow 30′
Count	2,903	2,431	472	442	30
($) MC Sum	26,217,754,755,404	6,177,292,648,268	20,040,462,107,136	14,303,673,004,544	5,736,789,102,592
($) MC Min	21,050,850	21,050,850	1,244,719,232	1,244,719,232	34,774,474,752
($) MC Max	617,588,457,472	401,644,421,120	617,588,457,472	538,572,161,024	617,588,457,472

Table 4.3: Makeup of market indexes by number of constituents as of Q4 2016. Additionally, the Market Capitalization (MC) of the smallest and largest constituent for each index is displayed along with the sum of all constituent MCs.

Chapter 5

Application 1: Deep learning and the ALAD Algorithm

5.1 Deep Learning Models and The ALAD Algorithm

In this section we describe the ALAD algorithm, the underlying deep learning models it uses, and our evaluation and experimental methods. We also explicitly state our research hypotheses, as Hypotheses 1 and 2 below. We return to these hypotheses in Section 5.3 and discuss how our experimental results support or refute them.

5.1.1 Method Overview and Definitions

Our approach to anomaly-based intrusion detection is a two stage process similar to that of Kim et al. but differs substantially in implementation [68]. We implement

a full detection pipeline consisting of two main stages. The first stage models the system call language using deep neural networks trained exclusively on baseline data. The second stage performs anomaly prediction using the model(s) from the first stage as well as an anomaly classifier.

Trace Probability

The first stage in our pipeline is a system call language model, which specifies the probability distribution for the next system call in a sequence given all prior system calls in that sequence. If we have a system call trace $t = x_1, x_2, x_3, \ldots, x_n$, we can calculate the probability of the sequence occurring with equation 5.1.

$$p(t) = \prod_{i=1}^{n} p(x_i | x_{1:i-1}) \tag{5.1}$$

Recall that each event x_i is a system call as described in Section 4.1. Models trained exclusively with baseline data estimate this probability distribution for a host's normal operation. Thus, we can formally define a model $\mathcal{M}$ as a mapping from traces t to a probability (real number) value. Details of the neural network architectures used, and their training methodologies can be found in Sections Sections 5.1.2 and 5.1.4 respectively.

Trace-Level Anomaly Detection ($TLAD$)

The second stage in our pipeline uses the probabilities generated by the first stage to classify a trace as baseline or anomaly. Specifically, a model $\mathcal{M}$ trained on baseline sequences can be used to classify a trace t as anomalistic if it has low probability. Taking the negative log of $\mathcal{M}(t)$ (its *negative log-likelihood*) results in low values if t

is not anomalistic, and high values if it is. A standard approach (e.g. [68]) to anomaly detection sets a threshold θ and classifies a trace t as anomalistic if its negative log-likelihood exceeds the threshold. Formally, trace-level anomaly detection ($TLAD$) is defined as follows, given a model $\mathcal{M}$ and threshold θ:

$$TLAD(t) = \begin{cases} 1 \text{ if } -\log(\mathcal{M}(t)) > \theta \\ 0 \text{ otherwise} \end{cases}$$

Application-Level Anomaly Detection ($ALAD$)

A drawback of $TLAD$ is that it considers only a single process at a time, whereas attacks typically target *applications* and can impact multiple processes. We propose an algorithm that aggregates predictions for all processes associated with an application. As discussed above in Section 4.1, process traces are endowed with application meta-information in corpora, which we can use to group traces into sets A as described below in Section 5.1.5. Furthermore, there is nothing special about this meta-information, in particular it is easily available to any system in practice. These sets A can be provided as input to our $ALAD$ algorithm to predict whether an application is benign or malicious. Formally:

$$\begin{aligned} ALAD(\mathrm{A}) = \;& \text{let } \{t_1, \ldots, t_n\} = \mathrm{A} \\ & \text{let } m = \text{median}(-\log \mathcal{M}(t_1), \ldots, -\log \mathcal{M}(t_n)) \\ & 1 \text{ if } m > \theta \text{ otherwise } 0 \end{aligned}$$

Figure 5.1 illustrates our complete pipeline using $ALAD$.

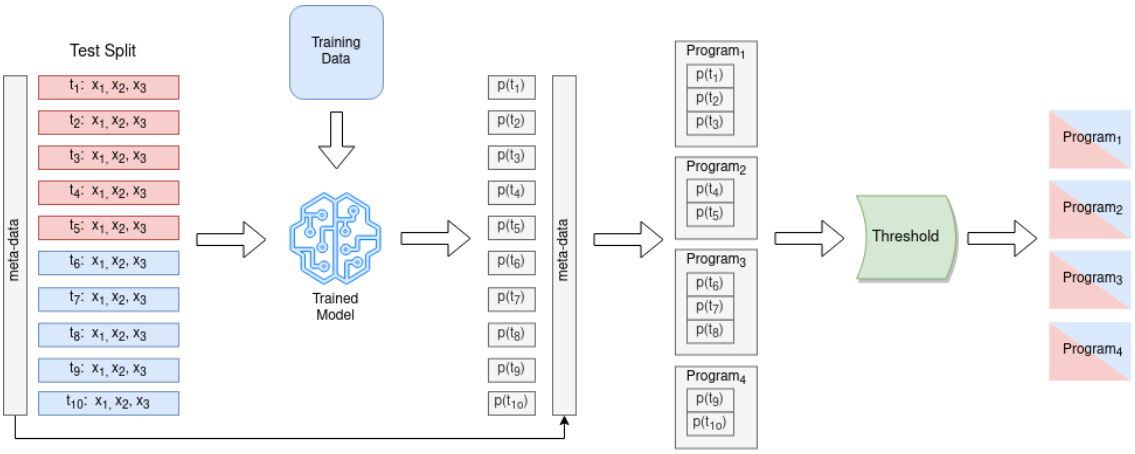


Figure 5.1: An illustration of our entire pipeline. Starting on the left is a testing split consisting of attack (red) and baseline (blue) system call traces. These are submitted to a model of normal behavior- the model is a result of training exclusively on baseline traces. The model is first used to obtain the probability of occurrence of each process trace in our test set. Then we use trace metadata to group trace probabilities by application. Finally, we test the aggregation (median) of these grouped probabilities against a threshold θ resulting in a classification for each program.

Research Hypotheses

With the above definitions in place, we can now state our explicit research hypotheses as follows.

Hypothesis 1 *WaveNet will outperform the LSTM and combined CNN/RNN architectures used in prior work [68, 70].*

Hypothesis 2 *ALAD will outperform TLAD as an IDS mechanism.*

We discuss the performance metrics and evaluation methodology for both *TLAD* and *ALAD* in Section 5.1.5. In Section 5.2 we compare the performance of several models from each architecture (WaveNet, LSTM, CNN/RNN) and show how *ALAD* yields significant performance improvements compared to *TLAD*.

5.1.2 Model Architectures

Intrusion detection is a less-explored application for the machine learning community, though many advances in the field are relevant. In particular, if we formulate the anomaly-based IDS as sequence-to-sequence learning problem, then we can leverage cutting-edge techniques from an active area of research in the deep learning community. We investigate and compare several models that are adapted from recent deep learning research.

All models used in this work feature the same high level layout. The integer encoded system calls are fed into a learned embedding layer. The embedding layer is followed by one of the architectures described above which outputs a probability distribution over system calls at each time step.

Our first candidate model is the WaveNet architecture [118], an audio generation model developed by Google DeepMind. WaveNet can serve as a drop-in replacement for LSTM-based architectures, which are commonly used on sequence-to-sequence problems. WaveNet employs discrete convolutions to capture context information and inform predictions, rather than the recurrent connections seen in LSTMs. This allows WaveNet to achieve superior performance with shorter training time as compared to LSTM-based architectures.

Our second and third candidates replicate the architectures from two prior approaches performing anomaly detection on ADFA-LD. They are an LSTM language architecture from Kim et al. [68], and the combined CNN/RNN architecture from Chawla et al. [70]. The LSTM architecture is simply a variable number of LSTM layers followed by dropout leading into a dense layer. The combined CNN/RNN

model features multiple one dimension convolutional layers stacked on top of a GRU followed by a dense layer.

We implemented these architectures in Python using TensorFlow version 2.1 [153] and provide our source code on GitLab [132].

5.1.3 Data

We constructed separate training, testing and validation subsets for both ADFA-LD and PLAID. A separate dense integer encoding was used for each dataset as they were generated on machines using different instruction set architectures. The testing sets feature a 1:1 ratio of attack and normal traces while the training and validation sets contain exclusively normal traces.

ADFA-LD

The ADFA-LD data directory consists of three folders: attack, training and validation. Respectively, these contain 746, 833, and 4,373 system call traces of varying lengths. The 175 unique system calls in ADFA-LD originally represented by a sparse integer encoding are refactored into a dense encoding for computational efficiency. The training and validation folders contain traces of normal operation while the attack folder features all attack traces.

We use this data to construct our own training, testing, and validation splits as follows. The ADFA training and validation folders are merged, consolidating all normal traces. Our test set was created by combining the attack sequences with 746 randomly selected normal sequences resulting in a 1:1 ratio of attack and normal sequences. The unused normal traces were then randomly split into training and

validation sets with an 80:20 ratio resulting in 3,567 sequences selected for training and 892 for validation. Note that the original ADFA data split is not used in this paper and all further references to training, testing and validation refer to our own data splits.

PLAID

Pre-processing of the PLAID dataset was done similarly. PLAID consists of two top-level directories, attack and normal, named for the type of traces they contain. A total of 1,494 traces with a length less than 8 or greater than 4,495 were discarded. The remaining traces consisting of 228 unique system calls were encoded with a dense integer representation. These bounds correspond to the smallest and largest sequences present in ADFA-LD. The test set is constructed by combining all 1,145 remaining attack sequences with an equal number of randomly selected normal traces. The remaining unused normal traces are then randomly split into training and validation sets with an 80:20 ratio resulting in 29,626 sequences selected for training and 7,407 for validation.

Complexity

We note that size of these datasets may seem small for deep learning applications; this observation however fails to consider the size of the overall landscape. There are k^n possible system call traces of length n, where k is the vocabulary size of system calls. The Linux kernel currently features over 300 unique system calls resulting in over 27 million possibilities for traces of length three. With a length of 4,495 the landscape for the largest traces under consideration is much larger than the number

of floating-point operations the universe could have performed thus far [154]. Given the complex landscape of system call traces, it is unsurprising that deep learning is required to achieve state-of-the-art performance.

5.1.4 Model Training & Configuration

For each architecture described in Section 5.1.2, we build three models with differing hyper-parameters to be used in an ensemble. The models, written M_i for $i \in \{0, 1, 2\}$ and $M \in \{\text{CNN/RNN, LSTM, WaveNet}\}$, are ordered by increasing number of parameters.

Selecting optimal hyper-parameters is a notoriously difficult task due to the large search space and computational cost of exploration. We used a Gaussian process optimizer to inform the search, aiding in the selection of hyper-parameters for our WaveNet models [155]. Ultimately we selected three WaveNet configurations all with 8 WaveNet blocks and no regularization. The models differed only by the number of filters in each convolutional layer which were 128, 256, and 512 respectively.

For the replicated architectures we used the hyper-parameters specified in their respective papers. For the LSTM architecture this was a single LSTM layer with 200 cells, a single LSTM layer with 400 cells, and two LSTM layers with 400 cells. The CNN/RNN models differed in both the number of 1D convolutions 6, 7, 8 and number of GRU units 200, 500, 600 respectively. The number of filters in each convolutional layer was set to mach it's WaveNet counterpart as the value was unspecified in the original work.

We trained all of our models using the Adam optimizer [156] with a learning rate of 0.0001. Gradient clipping with a maximum norm of 5 was applied to ensure

training stability [157]. Models were trained for a fixed number of epochs, 300 and 30 for ADFA-LD and PLAID respectively with a batch size of 32. A differing number of training epochs were selected for ADFA-LD and PLAID as the latter contains over eight times for training data. By using both a fixed number of training epochs and batch size for all models we ensured they received the same number of gradient updates allowing for a fair architecture comparison. Sparse categorical cross-entropy was used as the loss function for all models. The number of parameters, training time, and other summary information for each model is detailed in Table 5.1.

5.1.5 ID Classifier Evaluation

We completed 540 evaluation trials over combinations of dataset, model, and replicate. The nine model configurations outlined in Section 5.1.4 were trained and evaluated for thirty replication trials on both ADFA-LD, and PLAID. Our evaluation compares the *ALAD* and *TLAD* classification algorithms using these underlying models.

Both PLAID and ADFA-LD group traces by attack trial, allowing us to aggregate traces at the application level. The ADFA-LD baseline data does not include program grouping information, so we randomly sampled synthetic programs of equal size from the normal portion of the test set. For sake of consistency, we use the same process on PLAID.

In practice, we bootstrapped the baseline groups with thirty trials for each replicate model. This mitigates statistical errors from the random sampling, such as selection of an unrepresentative grouping. Thus, the single value result is the mean of the bootstrapped operations.

By varying the threshold value θ we obtained Receiver Operating Characteristic

	Params.	Training Time (h:m:s)	Eval. Time (s)	AUC *TLAD*	FPR *TLAD* (TPR = 1)	AUC *ALAD*	FPR *ALAD* (TPR = 1)
ADFA							
CNN/RNN_0	552096	1:41:55 ± 2:29	29.6 ± 0.9	0.785 ± 0.006	0.843 ± 0.030	$0.981^{\dagger}$ ± 0.003	$0.085^{\dagger}$ ± 0.014
CNN/RNN_1	2528472	2:48:15 ± 2:14	29.5 ± 0.8	0.802 ± 0.005	0.863 ± 0.076	$0.985^{\dagger}$ ± 0.002	$0.112^{\dagger}$ ± 0.037
CNN/RNN_2	7841280	4:53:42 ± 3:25	33.1 ± 4.9	0.800 ± 0.007	0.887 ± 0.082	$0.986^{\dagger}$ ± 0.002	$0.120^{\dagger}$ ± 0.055
$LSTM_0$	391376	1:43:23 ± 2:56	**27.0** ± 0.8	0.726 ± 0.013	0.962 ± 0.068	$0.924^{\dagger}$ ± 0.013	$0.255^{\dagger}$ ± 0.060
$LSTM_1$	1422576	2:50:30 ± 3:08	27.3 ± 0.9	0.759 ± 0.017	0.873 ± 0.070	$0.964^{\dagger}$ ± 0.015	$0.118^{\dagger}$ ± 0.044
$LSTM_2$	2704176	4:36:27 ± 5:05	45.8 ± 0.5	0.793 ± 0.005	**0.795** ± 0.009	$0.983^{\dagger}$ ± 0.002	$0.074^{\dagger}$ ± 0.010
$WaveNet_0$	1111664	**1:19:33** ± 0:48	39.3 ± 3.7	0.815 ± 0.004	0.795 ± 0.050	$0.986^{\dagger}$ ± 0.001	$0.144^{\dagger}$ ± 0.062
$WaveNet_1$	4346736	2:58:54 ± 0:59	38.5 ± 3.3	**0.830** ± 0.007	0.827 ± 0.038	$\mathbf{0.993}^{\dagger}$ ± 0.001	$\mathbf{0.036}^{\dagger}$ ± 0.008
$WaveNet_2$	17206640	8:15:56 ± 3:22	45.9 ± 6.8	0.828 ± 0.017	0.837 ± 0.047	$0.993^{\dagger}$ ± 0.004	$0.048^{\dagger}$ ± 0.065
PLAID							
CNN/RNN_0	569533	1:02:58 ± 1:06	45.7 ± 7.4	0.854 ± 0.024	0.719 ± 0.209	$0.980^{\dagger}$ ± 0.009	$0.220^{\dagger}$ ± 0.189
CNN/RNN_1	2561809	1:41:30 ± 1:36	47.2 ± 3.9	0.844 ± 0.030	0.625 ± 0.147	$0.970^{\dagger}$ ± 0.017	$0.248^{\dagger}$ ± 0.199
CNN/RNN_2	7879917	2:54:41 ± 2:06	48.9 ± 5.4	0.810 ± 0.029	0.683 ± 0.143	$0.945^{\dagger}$ ± 0.039	$0.312^{\dagger}$ ± 0.161
$LSTM_0$	412629	1:01:48 ± 1:34	39.2 ± 4.0	0.886 ± 0.008	0.543 ± 0.096	$\mathbf{0.985}^{\dagger}$ ± 0.004	$\mathbf{0.185}^{\dagger}$ ± 0.056
$LSTM_1$	1465029	1:41:17 ± 2:33	**39.1** ± 6.0	0.883 ± 0.060	0.572 ± 0.136	$0.968^{\dagger}$ ± 0.097	$0.254^{\dagger}$ ± 0.169
$LSTM_2$	2746629	2:42:03 ± 3:28	67.7 ± 4.8	**0.889** ± 0.011	**0.459** ± 0.117	$0.985^{\dagger}$ ± 0.006	$0.198^{\dagger}$ ± 0.135
$WaveNet_0$	1120409	**0:51:48** ± 0:41	68.4 ± 13.8	0.796 ± 0.036	0.661 ± 0.143	$0.936^{\dagger}$ ± 0.046	$0.428^{\dagger}$ ± 0.241
$WaveNet_1$	4362265	1:51:19 ± 0:55	79.4 ± 15.1	0.772 ± 0.024	0.711 ± 0.172	$0.915^{\dagger}$ ± 0.039	0.558 ± 0.202
$WaveNet_2$	17235737	5:01:33 ± 2:35	93.2 ± 20.3	0.798 ± 0.079	0.660 ± 0.142	$0.922^{\dagger}$ ± 0.125	0.523 ± 0.296

Table 5.1: We note that our proposed classification methodology results in a significantly higher AUC for all models under consideration. All models were trained and evaluated on a NVIDIA Tesla V100 with 32GB VRAM provided by the Vermont Advanced Computing Core. Training and performance metrics above are reported as the mean of thirty trials ± one standard deviation. In total this table summarizes the results of 540 training and evaluation trials. Total training time for the 540 models, not including hyper-parameter tuning, was over 62 days. We the relative efficiency of WaveNet whose smallest configuration had the fastest training time despite having over twice the parameters of the smallest model. ALAD performance metrics marked with † are statistically distinct (two-sided t-test, $p < 0.001$) from their TLAD counterpart. Evaluation time is how long it took the model to output the probability distribution for all sequences in the test set. Bolded results are the best in their respective column, and dataset combination.

	AUC *TLAD*	FPR *TLAD* (TPR = 1)	AUC *ALAD*	FPR *ALAD* (TPR = 1)
ADFA				
Avg. CNN/RNN	0.800 ± 0.004	0.842 ± 0.030	$0.985^{\dagger} \pm 0.002$	$0.125^{\dagger} \pm 0.027$
ReLU. CNN/RNN	0.800 ± 0.004	0.847 ± 0.041	$0.985^{\dagger} \pm 0.002$	$0.131^{\dagger} \pm 0.041$
Avg. LSTM	0.765 ± 0.006	0.903 ± 0.079	$0.966^{\dagger} \pm 0.006$	$0.228^{\dagger} \pm 0.030$
ReLU. LSTM	0.766 ± 0.005	0.903 ± 0.079	$0.966^{\dagger} \pm 0.006$	$0.231^{\dagger} \pm 0.029$
Avg. WaveNet	0.870 ± 0.008	0.712 ± 0.071	$\mathbf{0.998}^{\dagger} \pm 0.001$	$\mathbf{0.026}^{\dagger} \pm 0.005$
ReLU. WaveNet	$\mathbf{0.871} \pm 0.008$	0.692 ± 0.079	$0.998^{\dagger} \pm 0.001$	$0.027^{\dagger} \pm 0.005$
Hybrid_0	0.800 ± 0.005	0.661 ± 0.023	$0.975^{\dagger} \pm 0.004$	$0.153^{\dagger} \pm 0.030$
ReLU. Hybrid_0	0.801 ± 0.005	0.543 ± 0.050	$0.976^{\dagger} \pm 0.003$	$0.150^{\dagger} \pm 0.030$
Hybrid_1	0.820 ± 0.009	0.609 ± 0.017	$0.981^{\dagger} \pm 0.007$	$0.098^{\dagger} \pm 0.039$
ReLU. Hybrid_1	0.822 ± 0.009	0.504 ± 0.019	$0.981^{\dagger} \pm 0.007$	$0.100^{\dagger} \pm 0.037$
Hybrid_2	0.847 ± 0.005	0.547 ± 0.029	$0.990^{\dagger} \pm 0.002$	$0.047^{\dagger} \pm 0.008$
ReLU. Hybrid_2	0.848 ± 0.005	$\mathbf{0.485} \pm 0.034$	$0.990^{\dagger} \pm 0.002$	$0.047^{\dagger} \pm 0.008$
PLAID				
Avg. CNN/RNN	0.919 ± 0.012	0.499 ± 0.058	$0.993^{\dagger} \pm 0.004$	$0.119^{\dagger} \pm 0.042$
ReLU. CNN/RNN	0.919 ± 0.012	0.481 ± 0.050	$0.994^{\dagger} \pm 0.004$	$0.127^{\dagger} \pm 0.051$
Avg. LSTM	0.929 ± 0.020	0.394 ± 0.103	$0.994^{\dagger} \pm 0.009$	$0.099^{\dagger} \pm 0.141$
ReLU. LSTM	$\mathbf{0.930} \pm 0.012$	$\mathbf{0.380} \pm 0.098$	$0.995^{\dagger} \pm 0.006$	$0.098^{\dagger} \pm 0.140$
Avg. WaveNet	0.884 ± 0.055	0.559 ± 0.124	$0.977^{\dagger} \pm 0.055$	$0.197^{\dagger} \pm 0.135$
ReLU. WaveNet	0.886 ± 0.047	0.531 ± 0.058	$0.978^{\dagger} \pm 0.047$	$0.190^{\dagger} \pm 0.098$
Hybrid_0	0.929 ± 0.003	0.477 ± 0.076	$\mathbf{0.996}^{\dagger} \pm 0.001$	$\mathbf{0.063}^{\dagger} \pm 0.046$
ReLU. Hybrid_0	0.929 ± 0.003	0.466 ± 0.066	$0.996^{\dagger} \pm 0.001$	$0.065^{\dagger} \pm 0.046$
Hybrid_1	0.922 ± 0.037	0.512 ± 0.118	$0.989^{\dagger} \pm 0.034$	$0.113^{\dagger} \pm 0.165$
ReLU. Hybrid_1	0.923 ± 0.030	0.485 ± 0.066	$0.990^{\dagger} \pm 0.026$	$0.103^{\dagger} \pm 0.125$
Hybrid_2	0.914 ± 0.054	0.479 ± 0.120	$0.986^{\dagger} \pm 0.050$	$0.092^{\dagger} \pm 0.117$
ReLU. Hybrid_2	0.915 ± 0.049	0.459 ± 0.067	$0.986^{\dagger} \pm 0.048$	$0.089^{\dagger} \pm 0.102$

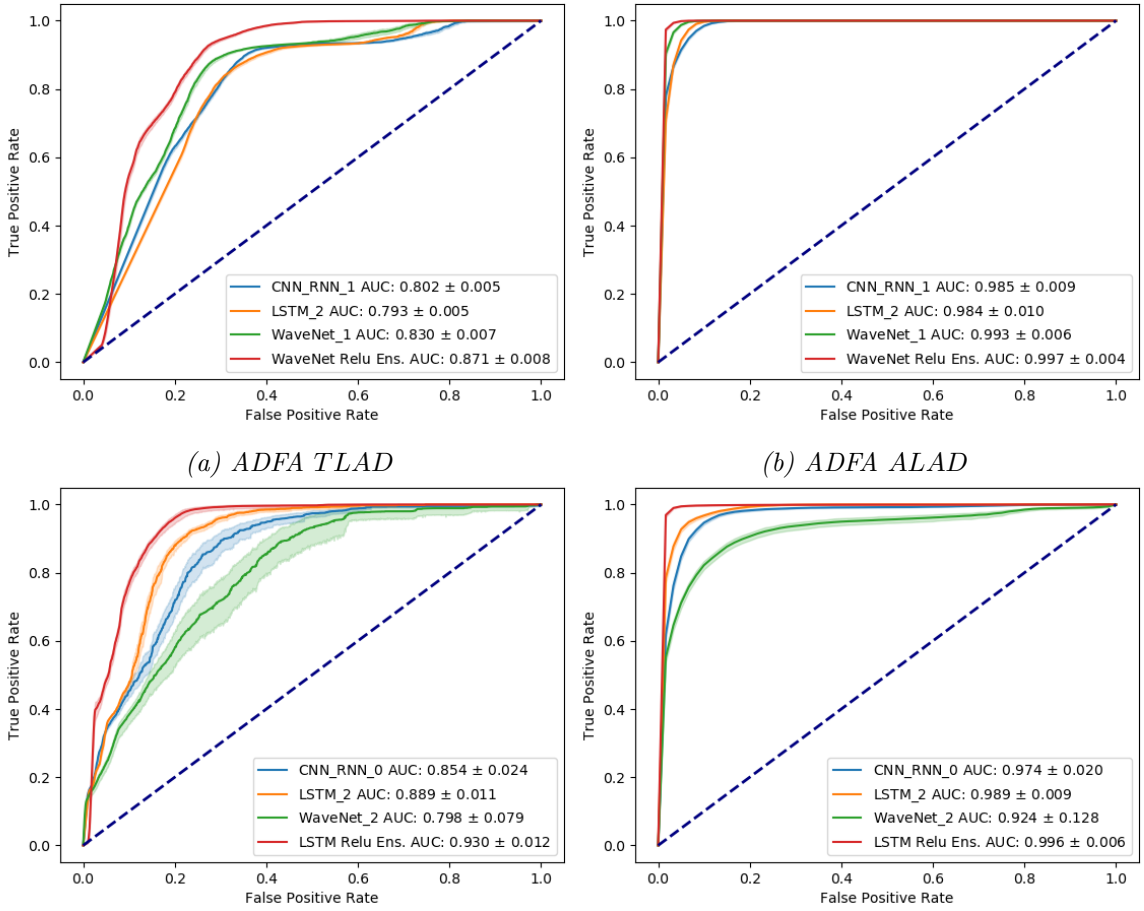
Table 5.2: Performance metrics for all ensembles under consideration. We note that ALAD results in a significantly higher AUC for all ensembles under consideration. Homogeneous ensembles, designated by architecture, contain all three model configurations from that architecture. Heterogeneous ensembles, termed hybrid, contain the the model from each architecture at the given configuration level. Performance metrics above are reported as the mean of thirty trials ± one standard deviation. ALAD performance metrics marked with † are statistically distinct (two-sided t-test, $p < 0.001$) from their TLAD counterpart. Bolded results are the best in their respective column, and dataset combination.

(ROC) curve for our classifiers—a common means of evaluating binary classification systems. The x-axis of the curve shows the false positive rate while the y-axis shows the true positive rate. In this case, the curve visualizes the trade-off between detection and false alarm rate. We summarize the performance of a model into a single value using the Area Under Curve (AUC) metric. In addition, we report the False Positive Rate (FPR) where the True Positive Rate (TPR) is one. The reported value for a given metric such as AUC (discussed below in Section 5.2) is the mean of all 30 replicate trials. For *ALAD* the reported AUC is the mean of 900 operations- thirty replicate trials each with thirty bootstrap groupings.

Finally, we also consider the same evaluation strategies for ensembles. We consider two ensemble types: a simple averaging, and the ReLU ensemble method from Kim et al. [68]. An ensemble of each type was constructed for each architecture and configuration level, resulting in 12 total ensembles. All ensembles consist of three models—either the three configurations from a given architecture, or the three different base models with the same configuration index.

5.2 Results

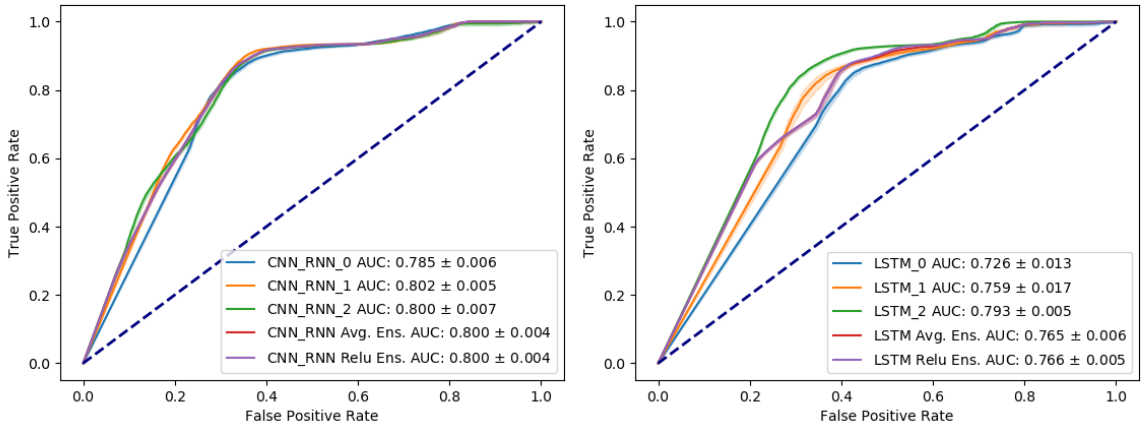
We present performance metrics, namely ROC AUC and FPR at complete detection for all models, in Table 5.1. Separate columns exist for both metrics over each combination of model, dataset, and classifier method. These metrics are reported as the mean of the thirty replicate trials $\pm$ one standard deviation. In all cases *ALAD* significantly increased AUC (two-sided t-test, p-val < 0.001) when compared to TLAD. We also observe a significant reduction in the FPR at complete detection in the vast



(a) ADFA TLAD *(b) ADFA ALAD*

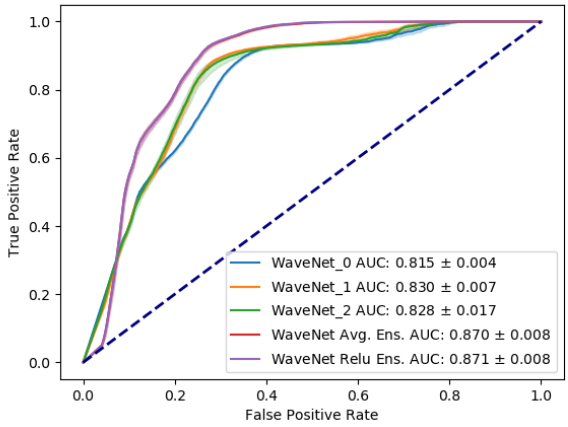
(c) PLAID TLAD *(d) PLAID ALAD*

Figure 5.2: ROC curves for the highest performing single model from each architecture along with the highest performing ensemble on ADFA (top) and PLAID (Bottom). Models were evaluated using both the TLAD(left) and ALAD(right). ROC curves show the mean and standard deviation for thirty trials. The legend reports the mean AUC and its standard deviation. For all models ALADsignificantly improved performance.

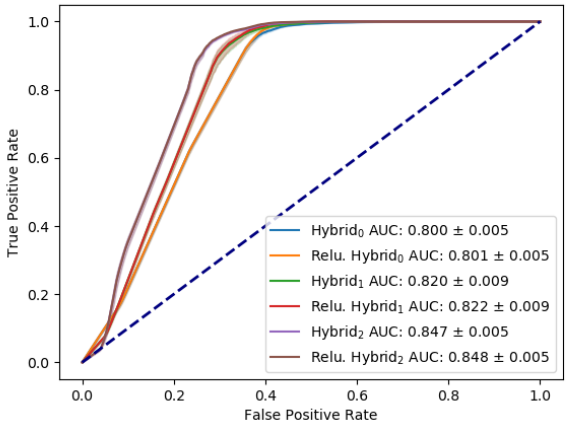


(a) CNN/RNN

(b) LSTM

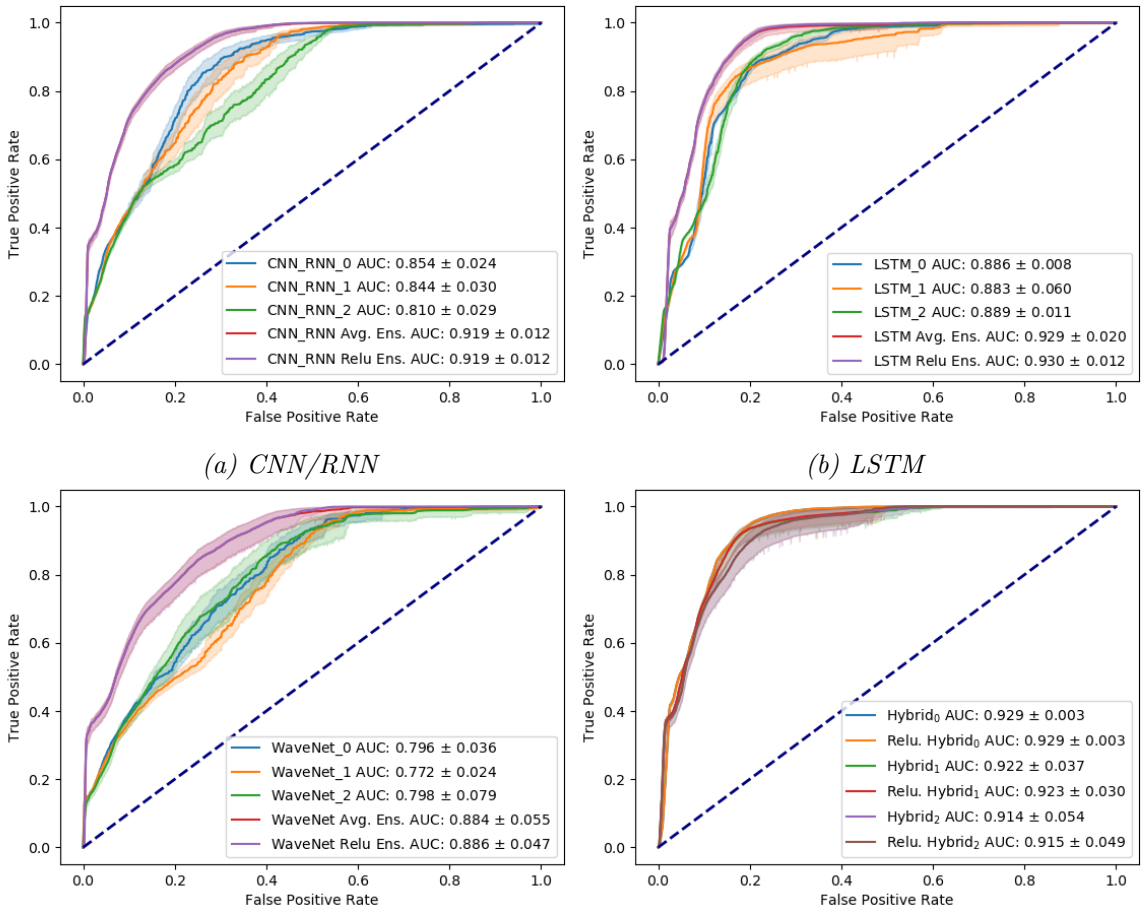


(c) WaveNet



(d) Hybrid

Figure 5.3: Figures 5.3a to 5.3c feature ROC curves for all trained models as well as homogenous ensembles on ADFA. Figure 5.3d shows the ROC heterogeneous ensembles constructed from model of all three architectures for each hyper-parameter configuration. ROC curves show the mean and standard deviation for thirty trials using TLAD. The legend reports the mean AUC and its standard deviation. We note that the LSTM and CNN/RNN ensembles under-performed some of their constituents while the WaveNet ensembles performed better.



(a) CNN/RNN *(b) LSTM*

(c) WaveNet *(d) Hybrid*

Figure 5.4: Figures 5.4a to 5.4c feature ROC curves for all trained models as well as homogenous ensembles on PLAID. Figure 5.4d shows the ROC heterogeneous ensembles constructed from model of all three architectures for each hyper-parameter configuration. ROC curves show the mean and standard deviation for thirty trials using TLAD. The legend reports the mean AUC and its standard deviation.

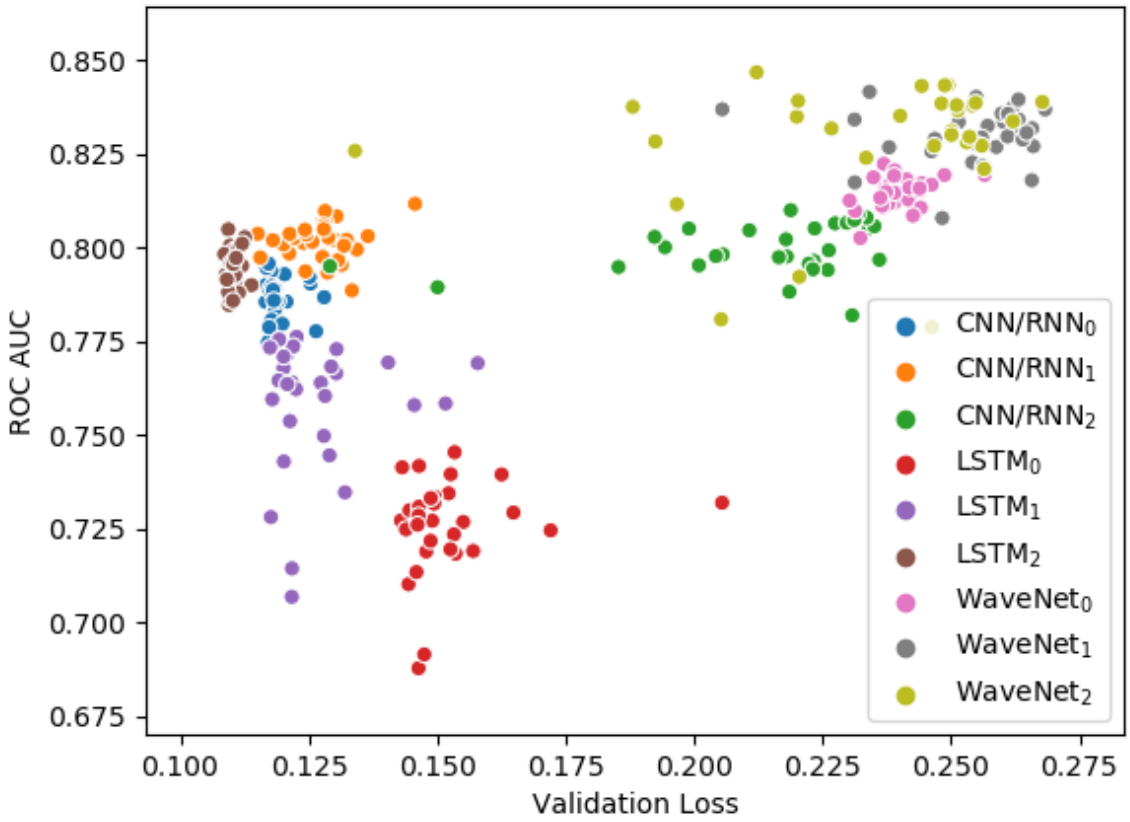


Figure 5.5: Validation loss compared to performance for all models on the ADFA dataset. Typically, one expects lower validation loss to correspond with higher performance. Here we see no strong correlation between validation loss and performance. We note that anomaly detection results in a special case as the training task (system call prediction), is not same as the evaluation task (attack classification).

majority of cases. WaveNet proved to be the strongest performer on ADFA while LSTM models had the strongest performance on PLAID.

In Figure 5.1.5 we show ROC curves for the highest performing model from each architecture, and the single best ensemble. We present our performance metrics for all 12 ensembles in Table 5.2. The traditional $TLAD$ is shown on the left, and our proposed $ALAD$ is on the right. We note the higher ROC curves when using $ALAD$ showing the lower false positive rates at all levels of detection. Of additional interest is that there is no clear winner in terms of model architecture or even model size. Models tended to have a higher performance on PLAID compared to ADFA-LD at the trace level, except WaveNet.

Figures 5.3 and 5.4 show ROC curves for all models and ensembles on ADFA-LD and PLAID respectively using $TLAD$. We use an identical evaluation methodology to Kim et al. [68] and Chawla et al. [70] at the trace level, so we would expect model performance to be similar to the original work despite the differing data splits and training methodology. This was the case for our CNN/RNN models which had AUCs similar to their originally reported values. We failed to replicate the high performance at the trace level of Kim et al [68], but our smaller LSTM model performed similar to the LSTM model used in Chawla et al. [70]. We did see a performance improvement from the use of ensembles and note that the ReLU ensemble was the top performer for both datasets, beating out the averaging and hybrid ensembles. Despite this we were unable to replicate the strong performance of the ReLU ensemble shown in Kim et al. [68] and note that its performance is virtually indistinguishable from the averaging ensemble.

In Figure 5.5 we compare validation loss at the final epoch to model performance

as measured by the ROC AUC score. One might expect a lower validation loss to correspond with a higher ROC AUC score, however we do not observe this empirically.

In summary, *ALAD* resulted in a significant AUC improvement for all models on all architectures and datasets under consideration. This improvement comes at virtually no additional computational overhead, compared to $TLAD$.

5.3 Discussion

5.3.1 Hypotheses

Testing our first of two hypotheses formulated in Section 5.1.1, namely that WaveNet would be the top performing architecture, produced mixed results. On ADFA, the dataset on which all models were tuned, WaveNet was indeed the top performer, supporting our hypothesis. However, WaveNet was the poorest performer on PLAID. There are two plausible explications for this behavior: WaveNet models may have over fit to the training data, or the architecture could be more sensitive to tuning.

Our second hypothesis, namely that *ALAD* would yield superior performance compared to $TLAD$, was fully supported by our analysis. For all models and datasets under consideration there was a statistically significant (two-sided t-test, p-val $<$ 0.001) improvement under *ALAD*. We speculated that this is due to the fact that some attack traces may in fact be benign. This is an unavoidable artifact of the collection methodology. The attack set contains all traces, each representing a distinct process, of a program during a successful attack. The effects of a modern attack are seen across multiple processes[50]. Precisely identifying the affected processes would

require knowing exactly what system calls would have been issued in the absence of an attack.

5.3.2 Practical Concerns & Use Cases

The information in Tables 5.1 and 5.2 allows practitioners considering a deep learning IDS deployment to make informed decisions about the trade-offs between detection, false alarms, and computational cost. These tables show the primary drawback of deep learning powered IDS, long training and non-trivial evaluation times. For real-time detection the time and computational requirements may be too expensive for some applications. However, in addition to real-time detection, IDSs may also be used in a retrospective analysis. In a retrospective analysis IDSs may be used to identify which systems or applications were affected; helping analysts identify the impact of a breach or informing their search.

While PLAID improves upon ADFA-LD there is still a need for more comprehensive datasets. To be effective IDSs must be trained on baseline data reflective of their host. To meet this requirement practitioners must train the systems they wish to deploy on data collected locally. Additionally, the system must be (at least partially) retrained when any significant changes occur, such as the deployment of a new application.

5.3.3 Implementation Decisions

A deployment of any form of anomaly detection requires practitioners to select a threshold θ. This is an obstacle for practitioners as there is no way to know a priori

the estimated probability the model will assign an attack sequence. Fortunately, there are two informed methods through which practitioners may select this value. First, one may use results on an existing corpus such as PLAID or ADFA. Second, one could utilize baseline sequences from their own production system; selecting a threshold that results in FPR they are able to handle. Of course, while neither of these choices guarantee complete detection they provide a means to achieve strong performance with an anomaly-based IDS. There is no wrong choice for a threshold value, only trade-offs between detection and false alarms.

Model selection is yet another obstacle for practitioners deploying a deep learning powered IDS. Typically, in deep learning one performs this task by selecting the model with the lowest validation loss. Unfortunately, we observed no strong correlation between AUC and validation loss. For this reason we recommend practitioners select their models based on their performance on reference datasets such as PLAID and ADFA. Additionally, this result underscores the need for researchers to continue to expand upon existing datasets.

Surprisingly, while we did see improvement from the use of ensemble, the effect was small compared to the performance achieved by the highest performing models. Additionally, while the ReLU ensembles outperformed their average ensemble counterparts, performance gains were marginal. As the creation of an ensemble requires duplicating training and evaluation costs, we believe it to be not worth the effort for this application.

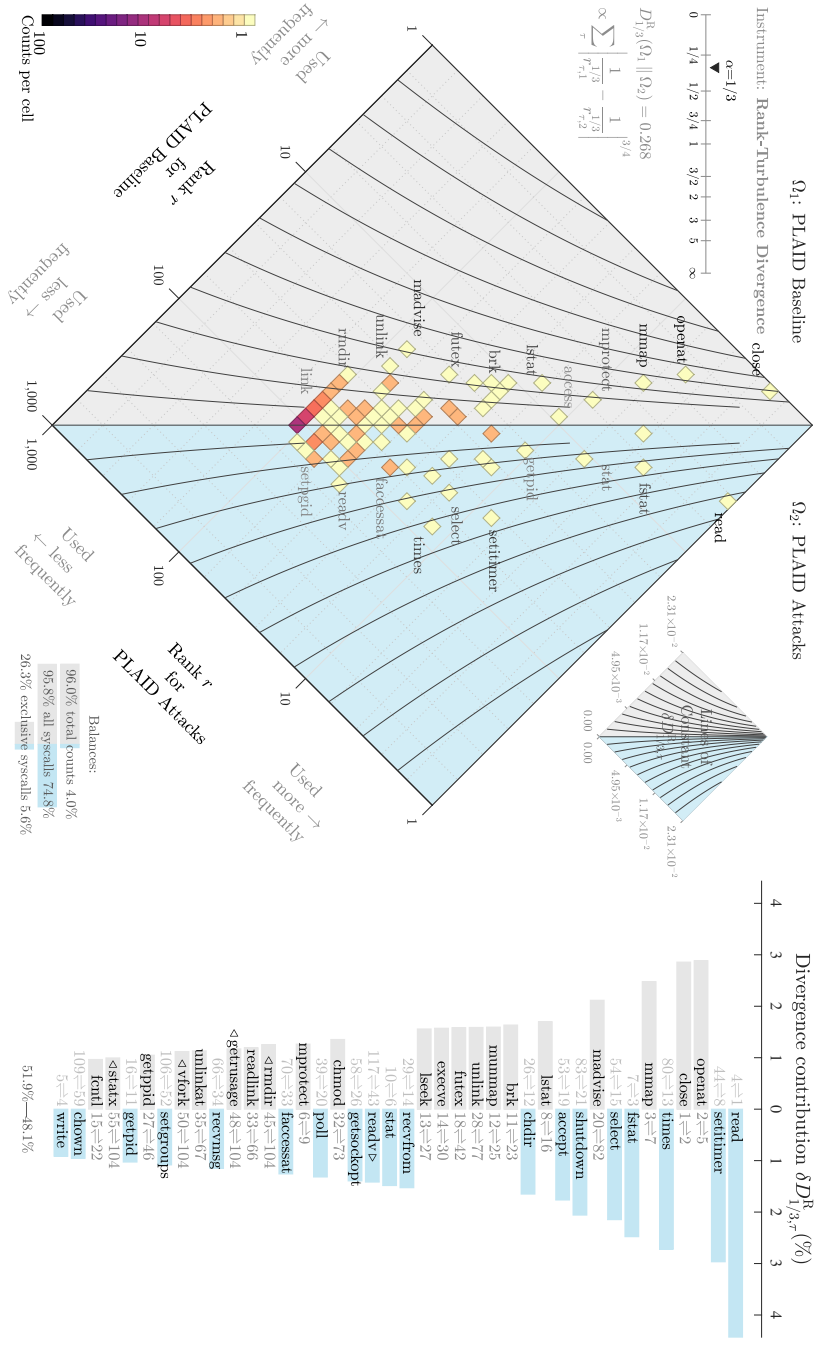


Figure 5.6: Comparison of system call rankings between attack and baseline traces in PLAID. Note that some of the most frequently utilized system calls, `read` *and* `close`*, are among the largest contributors to divergence.*

5.4 Visualizing Differences Between Baseline and Attacks

While deep learning is an effective ML approach in many applications, it suffers from its "black box", uninterpretable nature. Although methods are being developed to interpret deep learning models, they fall short, especially for insights into high-stakes decision making [158]. This is not *necessarily* an argument against the use of deep learning for HIDS since ML models are often just one component of a "observe, orient, decide, act" loop in security operation centers that also incorporate human analysts. However, interpreting data and predictive features in data is often critical for security practitioners. Instead of leveraging ML models for computational insights, we argue that other techniques can be leveraged, orthogonal to model development.

Two recently proposed techniques are "allotaxonometry" and "rank-turbulence divergence" [73]. These highly general methods leverage information-theoretic techniques for visualizing differences in datasets with complex structure, such as natural language text, baby names, and mortality cause databases. These techniques are especially relevant in our application space, since anomaly-based HIDS *rely* on the fact that significant differences exist between normal and malicious operations. Quantifying such differences not only sheds light on features potentially exploited by models, but also potentially new types of analysis. In this Section we explore the differences between attack and normal traces for both datasets used in this study, using allotaxonometry and rank-turbulence divergence.

In figure 5.6 we display the differences between attack and normal uni-grams using

an allotaxonograph. This instrument features a rank-turbulence histogram on the left, and a rank-turbulence divergence shift on the right. We compute the relative rate of usage for each uni-gram in the baseline and attack sequences separately, then order system calls using tied-rank. Ranks for system calls that are found in one distribution but not the other are replaced with the maximum rank of the joint distribution. The 2D histogram on the left displays the distribution of uni-grams found in the baseline and attack sequences as well as the overlap between the two distributions. System calls on the left side of the histogram are often used in the baseline sequences, whereas system calls that are highlighted on the right side of the histogram are often used in attack sequences. System calls which are used in both systems equivalently can be seen in the middle.

Of particular interest is that commonly used system calls (e.g., `open`, `close`, and `times`) display relatively high rank-turbulence divergence in both datasets. This is in contrast to natural language where rankings of the most common words tend to be stable across corpora [73]. Additionally, the most dangerous system calls [159] are not top contributors to divergence. This suggests that focusing exclusively on dangerous system calls could result in failures to detect intrusions. Additional allotaxonographs of uni through tri grams of both datasets are in Appendix A.1. We also contrast the raw frequencies of system calls found in baseline and attack traces for both datasets in Appendix A.2.

In figure 5.7 we show that system call usage roughly follows an exponential rank frequency distribution. The rank frequency system call bi and tri-grams appears to approximate a power-law with an exponential cutoff in the tail. Natural language corpora tend to be and stay power-law like for uni- through tri-grams with the tail

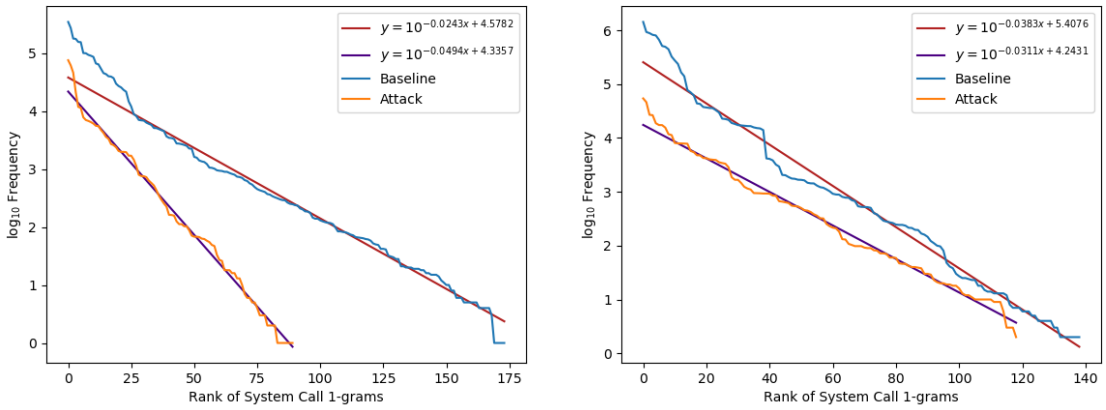


Figure 5.7: Rank frequency plots of system calls for attack and baseline traces in ADFA-LD (left) and PLAID (right). Fit lines were obtained using Huber regression. Observe that system call usage roughly follows an exponential rank frequency distribution. This is differs from natural language where word frequencies follow a power law distribution [1].

starting to flatten [1]. Thus, system call corpora become more power-law, while not quite reaching a power-law distribution while natural language corpora continue to follow a power-law distribution. Additional rank frequency plots for bi and trigrams are located in Appendix A.2. In all of these figures we clearly see substantial differences between attack and normal system call distributions.

Chapter 6

Application 2: Inefficiencies in the U.S. equity markets

Leveraging the large number of securities under study and the broad range of market capitalization (MC) covered, we examine scaling relationships between DSs, ROC, and MC. DSs occur in equities of all sizes. While DS are more frequent in equities with larger MC, the distributions of their qualities, such as their magnitude and duration, are more extreme among equities with smaller MC. We find a strong positive correlation between MC and ROC, show in Figure 6.1. A similar relationship is seen between MC-total trades and MC-differing trades in Figure A.9. The majority of ROC is generated by equities in the S&P 500 that are not also in the Dow (termed the SPexDow). The SPexDow also Granger-causes ROC in other mutually-exclusive market categories (Dow 30 and Russell 3000 less the S&P 500, or RexSP), pointing to its centrality in the U.S. equities market.

In the following sections, we describe statistics of DSs, including distributions of start times and durations. Next we move to analysis of ROC, providing summary statistics,

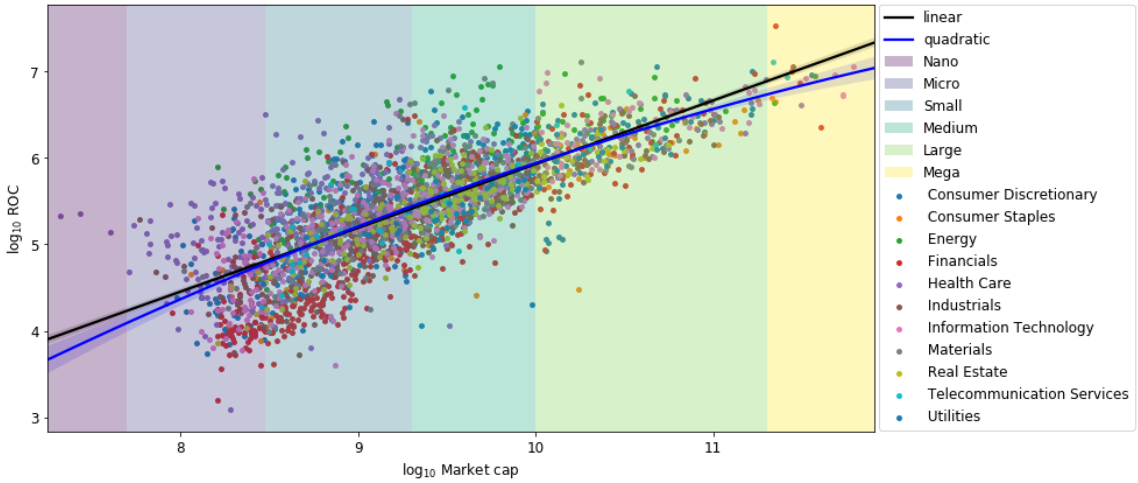


Figure 6.1: Linear and quadratic regression between Market Capitalization (MC) and ROC in doubly-logarithmic space. There is a strong positive relationship between MC and ROC. The data exhibits interesting nonlinearity and heteroskedasticity, where equities with smaller MC have higher variance in the dependent variable, while equities with larger MC have generally lower variance. Note that equities in the financial sector have a consistently lower ROC relative to MC while equities in the energy sector have a consistently higher ROC relative to MC. The shaded area surrounding the regression curves indicate 95% confidence intervals for the true curves, calculated using bootstrapping techniques.

comparisons across mutually-exclusive market categories, and correlation along with Granger-causality analyses. We close with a brief exploration of exchange-traded funds (ETFs), a discussion of results, and possibilities for future work.

6.1 Methods

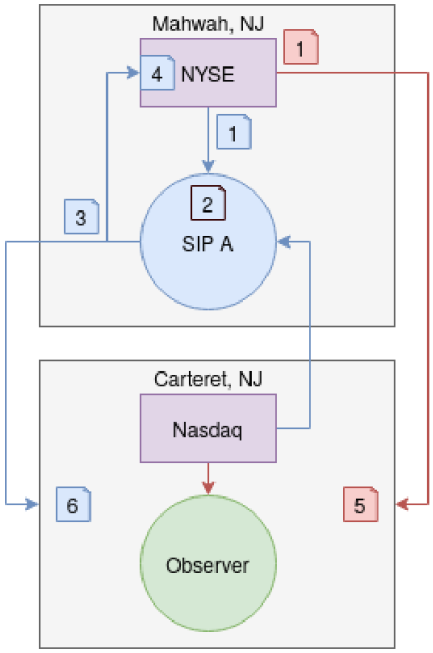
Our work investigates the occurrence of DSs and ROC arising from quote discrepancies between the SIP and direct feeds. Similar concepts have been discussed in empirical market microstructure literature [22, 78, 160, 161, 162], though formal definitions vary. We follow the definitions described below.

Dislocations Suppose that there exist two market data feeds, F_1 and F_2, each displaying quotes for a single asset. Quotes have the form $q_i(t) = (b_i(t), m_i(t), o_i(t), n_i(t))$, where $i \in 1, 2$, $b_i(t)$ is the bid price at time t, $o_i(t)$ is the offer priceat time t, $m_i(t)$ and $n_i(t)$ are the number of shares associated with the bid and offer at time t respectively. We observe these feeds from a single, fixed location in Carteret, NJ. A dislocation between these sources of data occurs when the prices of the quotes differ, e.g. $b_1(t) \neq b_2(t)$ or $o_1(t) \neq o_2(t)$. A DS occurs when the quotes differ and the relationship between the quoted prices remains constant, e.g. $b_1(t) < b_2(t)$ or $b_1(t) > b_2(t)$.

More formally we represent dislocation segments as a 4-tuple:

$$v_n = (t_n^{\text{start}},\ t_n^{\text{end}},\ \min \Delta p,\ \max \Delta p). \tag{6.1}$$

The maximum (resp. minimum) value of the dislocation segment are simply the max-



Step	Time (μs)	NYSE	Nasdaq\ Observer
0	0	9.98/10.00 9.98/10.00	9.98/10.00 9.98/10.00
1	0	9.98/10.00 **9.99/10.00**	9.98/10.00 9.98/10.00
2	5	9.98/10.00 9.99/10.00	9.98/10.00 9.98/10.00
3	97	9.98/10.00 9.99/10.00	9.98/10.00 9.98/10.00
4	102	**9.99/10.00** 9.99/10.00	9.98/10.00 9.98/10.00
5	282	9.99/10.00 9.99/10.00	9.98/10.00 **9.99/10.00**
6	379	9.99/10.00 9.99/10.00	**9.99/10.00** 9.99/10.00

Figure 6.2: We depict the dissemination of a market event to a subset of core participants in the national market system. The left panel visualizes the the plumbing connecting our participants; NYSE and SIP tape A co-located in Mahwah, NJ and Nasdaq along with our observer co-located in Carteret, NJ. All participants subscribe to both the SIP (blue) and direct feeds (red) from both exchanges. We show the flow of information as a sequence of enumerated events depicted as rectangular documents. The right panel displays the best bid and offer observed by the participants at each event from both the SIP (blue) and direct feeds (red). Note that while Nasdaq and our observer remain in sync for this entire example this is not always the case. We start at step zero with a market in harmony, that is all participants observe the same price on all feeds. Within the same microsecond NYSE processes an order resulting in a new best bid that narrows the spread. NYSE quickly dispatches a message of the top-of-book change to the SIP and its direct feed customers. Five microseconds later [2, 3] NYSE's message arrives at the SIP which takes an additional 92μs [4] to process the information and dispatch a new NBBO. After another five microseconds NYSE receives the new NBBO from its co-located SIP. It's not for another 180μs, 282μs after the original message the subscribers to NYSE's direct feed in Carteret receive the message. At this point we observe a 1¢ dislocation between the BBO displayed on the direct feeds and the observed NBBO. This dislocation persists for 97μs at which time the SIP update arrives in Carteret. Note that while technological advances will result in this sequence of events unfolding faster, the core behavior will remain unchanged. Messages from direct feeds travel a single leg, from exchange to subscriber, while updates to the NBBO require two legs, exchange to SIP to subscriber.

imum (resp. minimum) difference in the prices that are generating the dislocation segment over the time period $[t_n^{\text{start}}, t_n^{\text{end}})$. The time period $[t_n^{\text{start}}, t_n^{\text{end}})$ is determined by identifying a contiguous period of time where $\Delta p > 0$ or $\Delta p < 0$. From the above quantities the duration of the dislocation segment can also be calculated. The quantity $\Delta p(t)$ is the difference in the price displayed by the information feeds at time t as measured and timestamped by our observer in Carteret. From the definitions of $\max \Delta p$ and $\min \Delta p$ the reader will note that dislocation segments will tend to feature $\min(|\min \Delta p|) \geq \$0.01$, since the minimum tick size in the NMS is set at one penny for securities with a share price of at least $1.00. In collating dislocation data, we record the maximum and minimum value of each dislocation segment rather than a time-weighted average of dislocation value or other statistic for the sake of simplicity. In much of our analysis we take the absolute values of the maximum and minimum values of each dislocation segment as the fundamental object of study as any dislocation, regardless of which feed is favored, presents an opportunity for market inefficiency.

Figure 6.2 walks through an example DS occurring on a subset of the NMS using estimates of message transit and processing time for each leg of the journey. In our example a DS starts when a message regarding a quote change in Mahwah reaches our observer in Carteret via a direct feed and ends when the same message arrives via the SIP $92\mu s$ later. In this single example we see three factors that either alone, or in combination, may cause DSs; differences in processing time, transfer speed, and route (SIP messages require an additional leg). In this example the dislocation was triggered by a single top of book change at NYSE. However, dislocations can occur due to sequences of events occurring across multiple exchanges and SIP processors.

Recall that by definition a DS requires two feeds. TAQ data contains only the quotes resulting in a NBBO change as well as all trades. In contrast our dataset contains all quotes sent along the direct feeds as well as all SIP updates. Thus, we can observe events such as our example in Figure 6.2, an impossibility with TAQ data.

Realized Opportunity Cost Using the two market data feeds F_1, F_2 from dislocation definition above we calculate the ROC of using F_1 in place of F_2 by combining quote and trade information. Assume that trades take the form $T_j = (p_j, v_j, t_j)$, where p_j is the execution price, v_j is the number of traded shares, and t_j is the execution time. If a trade executes at one of the currently quoted prices, e.g. $b_1(t_j)$, then the ROC is given by $(b_2(t_j) - b_1(t_j)) * v_j$. If the trade executes on the opposite side of the book, e.g. $o_2(t_j)$, then the ROC is given by $(o_1(t_j) - o_2(t_j)) * v_j$. This allows for a consistent interpretation of the values, where a positive value indicates that F_2 displayed a better price for the active trader (higher bid or lower offer) than F_1. The total ROC over an interval $[S,\ E]$ is obtained by taking the sum of ROC values from all trades that occurred in that interval.

We first compute summary statistics and qualitative descriptions of the distributions of DSs and ROC. Additionally, we leverage the large sample of equities to conduct a cross-sectional study of the effect of company "size" on these microstructure quantities. We quantify the notion of size of a company by both its MC and its rank in relation to other companies. We also investigate index inclusion effects through the use of disjoint sets of equities and compute aggregate statistics across these sets. Since the S&P 500 is a strict superset of the Dow 30 and the Russell 3000 is a strict superset of the S&P 500, the natural division of the superset of all equities under

study is split into three distinct classes: the Dow 30, the S&P 500 excluding the Dow 30 (SPexDOW), and the Russell 3000 excluding the S&P 500 (RexSP). We investigate correlations between these disjoint subsets, and characterize the statistical properties of the time series of DSs and ROC across these disjoint categories. We further explore the relationship between these categories by conducting a Granger causality analysis of aggregated ROC time series [163].

The next section gives results on DSs, including summary statistics and regressions of DSs against MC. We then discuss structure in the intra-day distribution of DS start times and DS duration. Following this, we provide statistics of the ROC across the market as a whole and again within mutually-exclusive market categories. We then explore statistical properties of the ROC time series. We close with an overview of the statistics of ETF DSs and ROC, contrasting these with those of the market as a whole.

6.2 Results

6.2.1 Dislocation Segments

DSs can occur when quotes displayed by distinct information feeds differ. We cataloged all DSs occurring in the equities under study and present summary statistics along with qualitative comparisons of their distributions and higher-order moment statistics. Table A.1 - A.3 display means of summary statistics of DSs for each mutually-exclusive market category under study.

We will use the notation $\langle f_A \rangle$ to denote an average of the quantity f conditioned

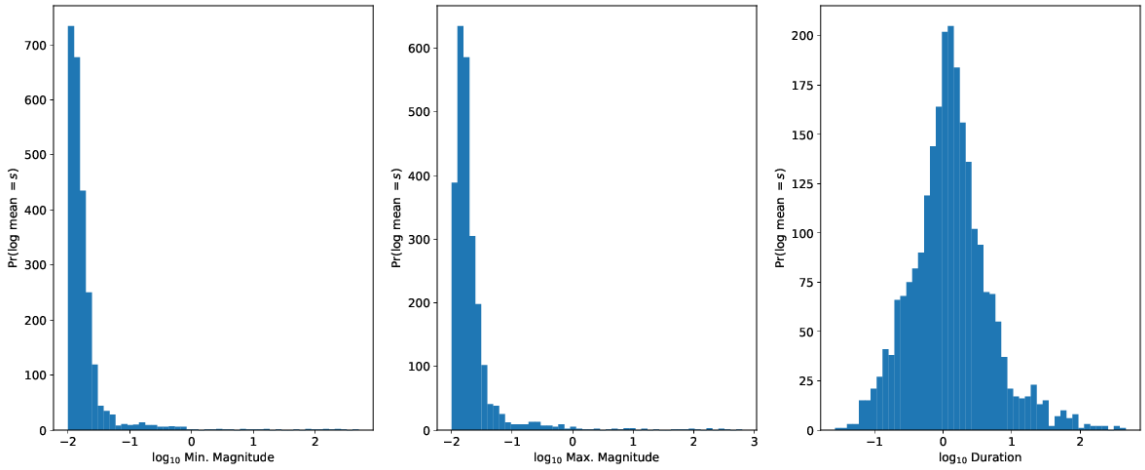


Figure 6.3: Histograms of the base-10 logarithm of minimum magnitude, maximum magnitude, and duration of dislocation segments in the RexSP without conditioning on duration or magnitude. The distributions are leptokurtic, with the log-distributions of minimum and maximum magnitude presenting a long right tail and the distribution of log-duration displaying a rough bell-shape.

on the condition A. These averages are interpreted as the quantity f conditioned on condition A averaged over all securities and all times of observation; defining the number of instances of the quantity f having condition A as N_A, we have

$$\langle f_A \rangle = \frac{1}{N_A} \sum_{\substack{1 \leq n \leq N_A \\ f \text{ has condition } A}} f_n. \tag{6.2}$$

Tables A.1 - A.3 show that, on average, there were more DSs in Dow 30 securities than in SPexDow or RexSP securities. However, the average maximum magnitude of DSs in the Dow30 is lower than those of the SPexDow, which in turn are lower than those of the RexSP. In particular, actionable DSs (those with duration $> 545\mu s$) with magnitude > \$0.01 exhibit more extreme behavior in the SPexDow and RexSP than in the Dow. On average, the median maximum magnitude in the Dow 30 among actionable DSs was $\left\langle \text{median max mag}_{duration,magnitude} \right\rangle \simeq$ \$0.023, while in

the SPexDow we observed $\left\langle \text{median max mag}_{duration,magnitude} \right\rangle \simeq \$0.034$ and in the RexSP $\left\langle \text{median max mag}_{duration,magnitude} \right\rangle \simeq \$0.045$, a roughly one-cent increase in the median maximum magnitude of a DS in each mutually-exclusive market category. Examples of distributions of these quantities are given in Figure 6.3, where the distributions of the means of minimum magnitude, maximum magnitude, and duration are plotted for the RexSP.

These results provide evidence for the existence of a MC scaling effect in DSs. Securities with larger MC tend to feature higher trading volume and more frequent occurrence of DSs, but these DSs tend to be smaller in magnitude on average. More frequent trading implies a lower probability that prices across differing information feeds will diverge by large magnitudes.

Since DSs are not distributed evenly throughout the day in the Dow 30 [26], we examine their distribution in the SPexDow and the RexSP as well. Appendix A.4 contains figures displaying the distribution of DS start times plotted modulo day and aggregated over the year as well as figures displaying the distribution of DS durations for each mutually exclusive market category. Distributions are plotted both without conditioning, conditioned on duration, as well as conditioned on duration and magnitude.

Distributions of start times display predictable structure. In all market categories, there are large peaks at the very beginning and end of the trading day (circa 9:30 AM and 4:00 PM), along with a noticeable and sudden increase in density around 2:00 PM. The peak in density that occurs at the end of the day is most noticeable when the distribution of start times is not conditioned on DS size. These observations correspond with the results found for the Dow 30 in [26]. However, along with

these granular observations, there also exists structure on shorter timescales. The distribution exhibits self-similarity on the half-hour timescale, with large peaks every half-hour and decreasing density toward a sudden peak at the next half-hour. There is also structure at the five-minute timescale that is noticeable before the 2:00 PM spike in density but does not appear to be present after the spike. (Future work could statistically test for the presence of this structure and for its persistence across multiple timescales.) The structure on shorter timescales is present in all distributions but, again, is more pronounced in distributions not conditioned on magnitude.

Distributions of DS duration are extremely heavy tailed, so we plot them with a log-transformed horizontal axis. All DS duration distributions exhibit one or more peaks in the range $10^{-4}s \leq \log_{10}\text{duration} \leq 10^{-3}s$, but there is also a distinct and much lower peak in the distribution near approximately one second in length.

S&P 500 Inclusion Effect: Dislocations As a visual aid to these results, we have included circle plots, as introduced in [26], to demonstrate the non-uniform distribution of DSs that can occur. Figure 6.4 shows these circle plots for two common stock pairs ((PBI, INCR), (BRK.B, XOM)) on the edges of our indices. The first pair is the smallest common stock in the S&P 500′ by MC that remained in the S&P 500 for the entire calender year and the closest component by MC in the RexSP, PBI and INCR respectively. The second pair is the only mega cap in the RexSP and the closest component by MC in the S&P 500′ that remained in the S&P 500 for the entire calender year, BRK.B and XOM respectively. We note that BRK.A is not included in the Russell 3000 [149] and that XOM is additionally included in the DOW. These common stock pairs underscore the difference in behavior between constituents of the

S&P 500 and those not included in the most worlds most widely tracked equity index.

Figure 6.4 displays the DSs for the above-mentioned common stocks aggregated over a year (modulo day). We see that DSs appear to be more concentrated for S&P 500 constituents with spikes occurring at the beginning of the trading day and at 2:00 pm. Additionally, DSs for S&P 500 constituents tend to have smaller magnitudes, relative to Russell 3000 constituents. We provide circle plots for many more securities on our webpage [164].

6.2.2 Market capitalization

Further evidence for scaling behavior arises from analysis of MC. Tables 4.2 and 4.1 display MC statistics broken down by industry sector and categorical size, e.g., micro-cap, mega-cap, etc. MC is significantly positively correlated with ROC. Tables A.13 - A.16 display results from ordinary least squares regressions predicting ROC using MC and other predictors. A linear fit predicting $\log_{10}$ *ROC* from $\log_{10}$ *MC*, $\log_{10}$ *total trades*, and $\log_{10}$ *differing trades* gives $R^2 \simeq 0.908$. A positive coefficient relates $\log_{10}$ *ROC* to $\log_{10}$ *MC*, indicating that higher MC is associated with higher ROC. A similar regression is computed including quadratic terms in $\log_{10}$ MC, which has a significant, but weak, negative association with ROC. Similar relationships hold for both the linear and quadratic models when the dependent variable is instead chosen to be total or differing trades.

Though behavior of ROC as a function of MC is generally similar when equities are stratified by sector, some sectors display lower average levels of ROC, differing trades,

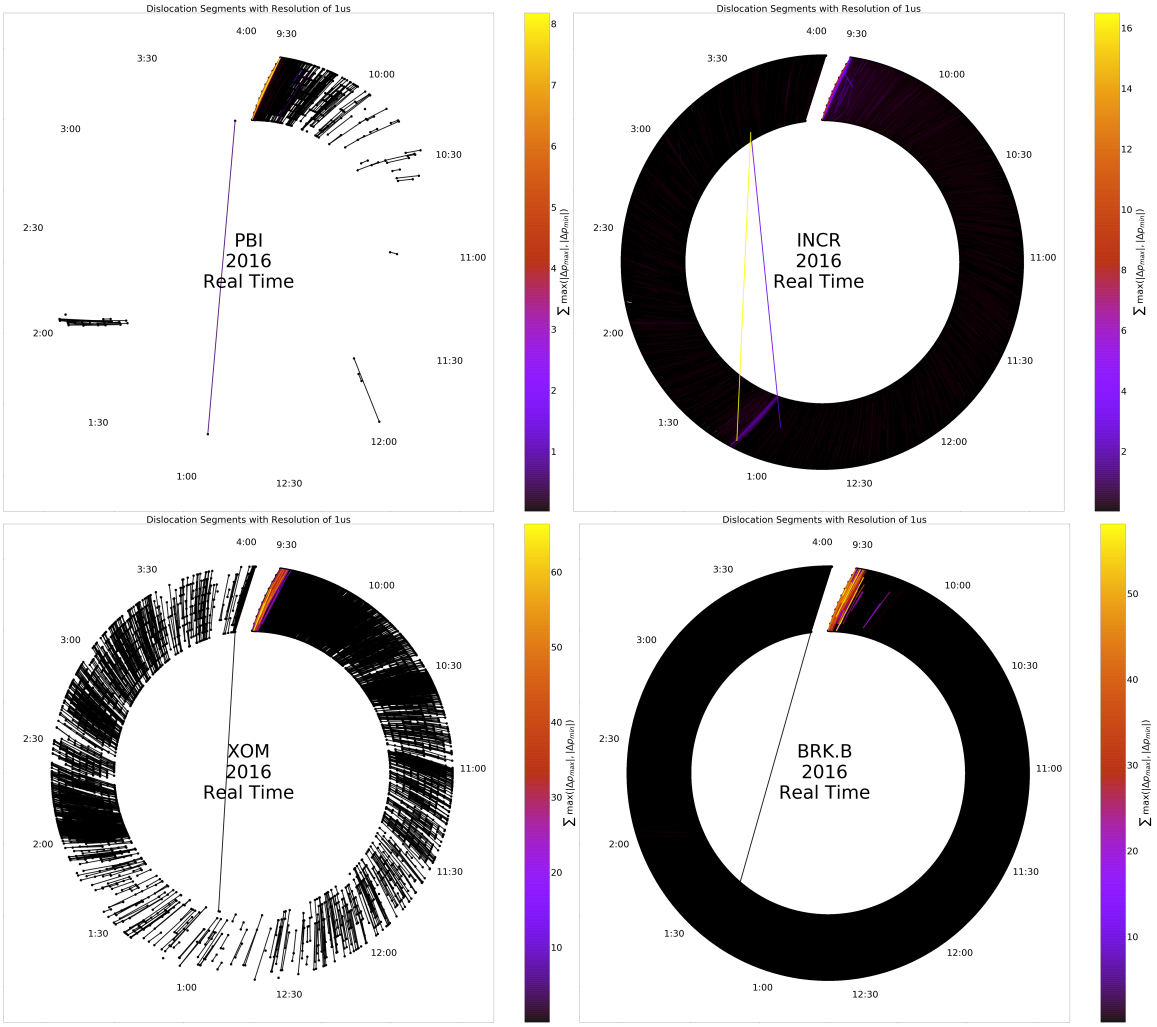


Figure 6.4: Dislocation segments (DS) for stock pairs (similar MC) aggregated over a year (modulo day). PBI (paired with INCR) is the smallest common stock by MC under consideration that remained in the S&P 500 for all of 2016. BRK.B (paired with XOM) is the only mega cap in the RexSP. We see that DSs appear to be more concentrated for S&P 500 constituents (left) with spikes occurring at the beginning of the trading day and at 2:00 pm. Additionally, we note that DSs appear to a smaller magnitude for S&P 500 constituents.

or total trades when MC is held constant. Equities classified as being in the financial sector generally have a smaller amount of ROC, while equities classified as being in the energy sector exhibit a higher amount of ROC on average. However, there is no clear general trend linking sectors to MC or to ROC.

6.2.3 Realized opportunity cost

As expected with an increase in the number of analyzed equities from 30 to more than 2900, the amount of ROC rose substantially from the quantity reported in [26], from $160M to $2.05B USD. ROC clearly displays sublinear scaling with the number of studied equities; we do not observe a thousandfold increase in the amount of ROC with a thousandfold increase in the number of equities. The information advantage afforded traders with access to direct feed information is not uniform; though a vast majority of the ROC ($1.91 B) favored the direct feeds in this way, a non-negligible amount of ROC ($137 M) did favor the SIP feeds. Approximately a quarter (23.71%) of all trades observed occurred during a dislocation. The fraction of "differing traded value"—the nominal market value of all differing trades—was slightly higher (25.25%) than the fraction of all trades that were differing trades. The ratio between these two values (25.25% / 23.71% = 1.0651) shows that the average differing trade moves approximately 6.51% more value than the average trade. This indicates a qualitative shift in trading behavior during dislocations.

Securities in the SPexDow account for a median of 2,006,091 differing trades per day, in contrast to the 309,158 in the Dow 30 or 1,921,121 in the RexSP. The median differing traded value per day in the SPexDow was also the highest among the three categories, totaling approximately $14.07T versus the RexSP's total of $6.7T and

	BRK.B	XOM	PBI	INCR
MC ($)	401,644,421,120	374,280,552,448	2,821,674,240	2,820,235,520
ROC ($)	2,278,835.98	8,846,416.18	726,596.69	487,049.13
Trades	5,120,595	16,146,652	2,360,470	904,613
Diff. Trades	1,544,050	4,479,209	488,092	243,855
Traded Val. ($)	70,435,832,686.71	169,057,336,872.77	5,766,285,837.56	3,989,174,661.59
Diff. Traded Val. ($)	24,162,015,573.13	47,541,675,580.93	1,257,265,907.34	1,016,834,174.82

Table 6.1: Summary statistics for select common stock pairs. BRK.B (paired with XOM) is the only mega cap in the RexSP. PBI (paired with INCR) is the smallest common stock by MC under consideration that remained in the S&P 500 for all of 2016. Note that those in the S&P 500 (green) have a much higher trading volume and ROC then their similarly capitalized counterparts.

the Dow's total of $3.27T. ROC per share differed across the three categories, with median ROC per share per day of 1.1¢, 1.5¢, 2.1¢for the Dow, SPexDow, and RexSP respectively. ROC per share tends to increase as MC decreases, with lowest ROC per share occurring in the Dow and highest ROC per share occurring in the RexSP. Median total ROC per day on the Dow amounted to $514.8K, while median total ROC per day on the SPexDow totaled $3.384M and on the RexSP amounted to $3.564M. Summary statistics for distributions of ROC for each mutually-exclusive market category are given in Table A.7.

It is interesting to consider the distribution of both total ROC and ROC per share by both equity and mutually-exclusive market category. Figure A.10 displays ROC of the top 30 and bottom 30 of all securities under study when ranked by ROC. Included in this figure for comparison is the exchange-traded fund SPY, an ETF that tracks the S&P 500. Selected ETFs are also treated separately in Section 6.2.4. It is notable that the equity with the largest ROC, Bank of America (BAC), has more than twice the ROC of the equity with the second-largest amount of ROC, Verizon (VZ). Though not an equity and not included in the rest of this study, it is also

Stat.	Mean	Std.
MC ($)	3,695,890,099.20 ±	464,930,329.63
	3,696,678,400.00 ±	465,021,263.08
ROC ($)†	1,530,766.70 ±	1,212,566.32
	573,704.19 ±	454,901.87
Trades†	3,757,345.30 ±	2,579,005.78
	1,340,988.30 ±	1,099,357.71
Diff. Trades†	848,648.80 ±	568,393.92
	318,163.00 ±	222,699.69
Traded Value ($)†	11,966,521,828.32 ±	6,995,211,619.34
	4,281,159,071.68 ±	2,466,969,453.45
Diff. Traded Value ($)†	2,930,334,696.21 ±	1,746,456,767.92
	1,071,563,501.93 ±	551,246,762.12

Table 6.2: Comparison of the smallest ten common stocks that remained in the S&P 500 for all of 2016 (green) and the ten RexSP common stocks with the closest MC. Rows marked with † have significantly (two-sided t-test, $p < 0.05$) higher values for common stocks in the S&P 500. We note that common stocks in the S&P 500 have nearly three times the trading activity and ROC than their similarly capitalized counterparts.

notable that SPY, one of the most heavily traded securities on the NMS along with BAC, is close to BAC in ROC. Of the top 30 securities with the most ROC, eight of the 30 are Dow 30 equities. Only four out of 30 are RexSP equities, while the other 17 non-ETF securities are SPexDow equities. One may attribute this to MC, though we note the S&P 500 is not the largest 500 U.S. companies 4.2. In fact, there are 612 RexSP constituents with a MC greater than PBI, a common stock at the bottom of the S&P 500. This includes 67 large and mega cap common stocks. Since the S&P 500 appears to be the primary driver of ROC across all equities (c.f. below), we find the top 30 and bottom 30 S&P 500 securities ranked by ROC, including Dow 30 securities, and plot their ROC in Figure A.11. Even in this subset, only 10 of the top 30 equities are Dow 30 securities. However, when the unit of analysis changes to ROC per share, as in Figure A.12, we find that RexSP equities fill 27 out of 30 top ranks, which corresponds with the aggregated statistics reported in Table A.7. Additionally, we revisit our common stock pairs from 6.2.1 to take a closer look at common stocks barely inside and outside of the S&P 500. We see that the common stocks in the S&P 500 have a much higher trading volume and ROC then their similarly capitalized counterparts 6.1. To see if this trend holds we expand our set to the ten smallest common stocks that remained in the S&P 500 for all of 2016 and the ten RexSP common stocks with the closest MC. None of the ten RexSP members spent any time in the S&P 500 during 2016. We find the trend holds with members of the S&P 500 having nearly three times the trading activity and ROC than their similarly capitalized counterparts 6.1.

Since there appear to be differences between the (stationary) summary statistics of the mutually-exclusive market categories, it is reasonable that there may be signifi-

cant differences between the ROC statistics considered as time-dependent stochastic processes and simply considered as random variables decoupled from time. Within each category, the ROC was computed for all equities in that category for each day. Each ROC series is then normalized as $r_i \mapsto \frac{r_i - \langle r_i \rangle}{\sqrt{\mathrm{Var}(r_i)}}$, which allows direct comparison of the series. Figure A.16 displays a quantile-quantile plot of the Dow, SPexDow, and RexSP ROC distributions. The Dow distribution is plotted as linear and the other two distributions are compared with it. It is immediately obvious that the left tails of the SPexDow and RexSP distributions are heavier than that of the Dow; this also appears to be the case for the right tails of the distributions, but there is little sampling in this region and so no conclusion can be drawn. This similarity of the SPexDow and RexSP distributions is also striking; when normalized they appear almost identical.

Figure A.17 displays the time-dependent sample paths of ROC sampled at daily resolution. These processes are anti-autocorrelated—they display mean reversion—as evidenced by their detrended fluctuation analysis (DFA) [165] exponents of $\alpha_{\mathrm{Dow}} = 0.438$, $\alpha_{\mathrm{SPexDow}} = 0.242$, and $\alpha_{\mathrm{RexSP}} = 0.235$. All series exhibit rare large values from time to time, with the Dow ROC series exhibiting the largest rare values relative to its mean fluctuations and the SPexDow series exhibiting the smallest. We also note that, in accordance with the QQ plot of the time-decoupled distributions above, the DFA exponents of the SPexDow and RexSP—and thus their corresponding dynamical behavior—are closer than they are to the Dow DFA exponent.

A review of the above results points to the SPexDow as being the “dominant" mutually-exclusive market category in some sense: it accounts for a plurality of differing trades, differing traded value, and total ROC, while also having a DFA exponent lower than

that of the Dow and close in value to that of the RexSP, meaning that its time-series of ROC is strongly mean-reverting. The amalgamation of these facts can be interpreted as evidence that the SPexDow ROC time series is possibly least likely to be influenced by the other series of ROC. To test this hypothesis, we conduct a number of Granger causality tests on the time series of ROC. Granger causality is the notion that past values of one time series may be useful in predicting current and future values of another time series [163]. A maximum lag of 40 days was set and four tests were calculated pairwise between each of the three mutually-exclusive market categories: sum of squared residuals χ^2-test, a likelihood ratio test, sum of squared residuals F-test, and a Wald test. We consider there to be a significant Granger causality between series when all four tests indicate significant Granger causality at the $p = 0.05/N_{\text{lags}}$ confidence level. The correction for multiple comparisons is done using the most conservative estimate, the Bonferroni correction, to minimize the probability of Type I error [166]. Figure 6.5 displays the results of these tests graphically as a directed network. The direction of edges denotes the direction of the Granger-causal relationship between the categories, while the weights on the edges denote the total number of lags for which the relationship was significant. The SPexDow is shown to significantly influence both the Dow and RexSP while not being significantly influenced by either category; this provides strong evidence to support our above hypothesis. We note that the SPY tracks the S&P 500, is one of the most heavily-traded securities, and has the second-highest amount of ROC of the securities under study here. The SPY's price dynamics and ROC may thus have a material effect on the relationships between the S&P 500's ROC and those of the other market categories, providing a partial confounding effect to the Granger-causal relationship determined here; there

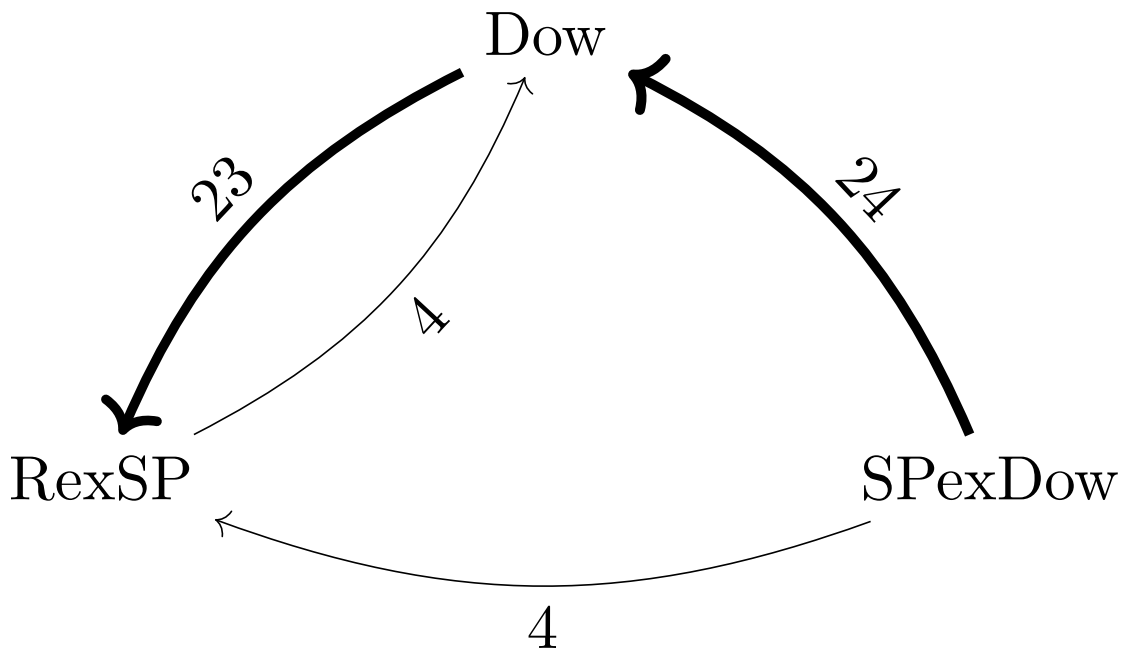


Figure 6.5: Network of relationships between mutually-exclusive market categories implied by results of four Granger causality tests. The direction of the edges gives the direction of the Granger-causal relationship, while the weight on the edge is the total number of lags for which the relationship was significant at the $p = 0.05/N_{lags}$ level (the conservative Bonferroni correction). The maximum number of lags was chosen to be $N_{lags} = 40$. Thickness of the edge is proportional to edge weight and is plotted for emphasis in visualization. Details about which lags were associated with significant Granger causality can be found in Table A.12.

may be a mutually-causal relationship between the real S&P 500 and the ETF that tracks it. The RexSP and Dow have a mutually Granger-causal relationship, with the Dow exerting more influence on the RexSP than the other way around. This finding corresponds with the ranking of categories on a total shares traded per number of equities basis; this is not a surprising result. We also find that the SPexDow exerts far less influence on the RexSP than does the Dow (four total lags for the SPexDow versus 23 total lags for the Dow), a fact for which we do not have a ready explanation.

ROC:	Dow	SPexDow	RexSP
Dow	1.000000	0.451072	0.319018
SPexDow	0.451072	1.000000	0.724903
RexSP	0.319018	0.724903	1.000000

ROC / Share:	Dow	SPexDow	RexSP
Dow	1.000000	0.103061	-0.019662
SPexDow	0.103067	1.000000	0.411443
RexSP	-0.019662	0.411443	1.000000

Table 6.3: Pearson correlation matrices of mutually-exclusive market categories. For each index subset a daily resolution time series is constructed for the given statistic over all stocks in the index subset. For the ROC series the ROC generated for each stock on a particular trading day is summed, while in the ROC per share case the values are averaged. The correlation coefficients are then calculated between pairs of time series in order to construct the tables above. The top table displays ROC correlations, while the bottom table displays ROC per share correlations. The ROC per share statistic normalizes the number of traded shares, allowing for a fair comparison between the more heavily traded stocks in the Dow 30 or S&P 500 subset with the more lightly traded stocks in the Russell 3000 subset.

Providing further evidence for the above hypothesis, we compute Pearson correlations between pairs of mutually exclusive categories for both ROC and ROC per share; these results are displayed in Table 6.3.

ROC correlations are strongest between SPexDow and RexSP ($\rho = 0.72$) and SPexDow and Dow ($\rho = 0.45$), while the correlation between the RexSP and Dow is lower ($\rho = 0.31$). ROC per share correlations are universally lower than those for ROC, but the correlations between SPexDow and RexSP ($\rho = 0.41$) and SPexDow and Dow ($\rho = 0.10$) are still higher than that between RexSP and Dow ($\rho = -0.01$), which is actually negative.

Figure A.13 displays the distributions of daily total ROC in 2016 by mutually-exclusive market category. The panel with linear scaling highlights the extremely heavy-tailed nature of these distributions, while the log scaled panel provides a bet-

ter comparison between the mutually-exclusive market categories. On average, members of the Dow 30 experience the greatest daily ROC, followed by members of the SPexDow, followed by members of the RexSP. It seems likely that the kurtosis of the theoretical distributions do not exist, implying tail exponent $\gamma < 4$ in the distribution $\Pr(X > x) \sim x^{-(\gamma-1)}$. Table A.8 displays the skew and kurtosis for each distribution. If we examine the daily ROC per share in a similar manner, which is shown in Figure A.14, we observe a reversal of the previous relationship. Members of the Dow 30 have the least daily ROC per share, on average, and members of the RexSP have the most. Though there is a slight trend, more ROC per share in less frequently traded stocks, the distributions of all three groups are nearly centered at 1¢ per share. This corresponds well with our expectations based on the structure of the system and the distribution of DS magnitudes shown in Figure 6.3.

6.2.4 ETFs

Exchange traded funds (ETFs) are securities that trade on the NMS and are designed to mimic as closely as possible a particular portfolio of other securities. They are thus governed by the same price discovery mechanism as other securities that trade on the NMS, as opposed to the end-of-day price discovery mechanism to which mutual funds are subjected, but also allow investors to own a portion of potentially many underlying assets (or at least a simulacrum of such), similar to a mutual fund. Here, we briefly remark on the similarities and differences between ETFs designed to track subsets of the market and those subsets of the market themselves. We calculate statistics on the DSs and ROC associated to ETFs from three firms (Vanguard, iShares, Russell) for three indices (S&P 500, Russell 300, Russell 2000), for a total of nine ETFs

(SPY, VOO, IVV, THRK, VTHR, IWV, TWOK, VTWO, IWN). The Russell 2000 is comprised of the smallest 2000 firms in the Russell 3000 by MC. The S&P 500 and Russell 3000 were selected as measures of overall market activity while the Russell 2000 was selected to isolate dynamics among ETFs that track smaller equities.

Table A.9 summarizes ROC statistics for the ETFs under study. The fraction of differing trades and differing traded value are lower than for any of the indexes as a whole; in fact, the ratio of the fraction of differing traded value to the fraction of differing trades is less than one. Total ROC incurred from trades in ETFs studied here totaled over $38 million in calendar year 2016. This statistic provides some evidence to suggest that ETFs have their own endogenous statistical behavior that differs from the behavior of the assets from which they are derived.

Chapter 7

Concluding Remarks

This dissertation has developed new quantitative methods for exploring the behavior of two computational systems. In the first application we developed new methods for host-based intrusion detection systems (HIDS). In the second application we showed that market inefficiencies in the form of dislocations and realized opportunity cost were common and of non-negligible frequency and size. These results are used to establish baselines for their respective systems; allowing practitioners, by they security personnel, investors, or regulators to better understand and evaluate the state of the system they are operating in.

7.1 Host-Based Intrusion Detection

Our fundamental approach to intrusion detection is to develop models for predicting "normal" aka baseline behavior, and then leveraging those models to detect malicious behavior as anomalistic. This approach has the benefit of being able to detect novel attacks, as well as known ones. We used deep learning models to achieve high levels

of prediction performance.

Our work makes four primary contributions in the area of HIDS research. First, we collected and publicly released PLAID, a new system-call dataset for developing and evaluating IDS. Second, we developed *ALAD* (*A*pplication-*L*evel *A*nomaly *D*etection), a new classification method for anomaly-based IDS. Third, we presented the largest comparison to date of deep learning architectures applied to this domain. Fourth, we explored new visualization methods, based on information-theoretic corpus divergence measures, for exploring HIDS datasets.

Evaluating the performance of advanced methods, such as alternative deep learning models, requires comprehensive benchmarking that cannot be accomplished with the use of a single dataset. In our own architecture comparison, the use of either PLAID or ADFA-LD independently might lead to a conclusive answer that is different from the relatively inconclusive results that we observed during a comprehensive evaluation. By introducing PLAID, we hope to empower the community to better evaluate new and existing HIDS models.

ALAD offered significantly better performance than *TLAD* regardless of the selected deep learning architecture or training dataset. This indicates that the inclusion of a relatively minimal piece of meta-data, application-level labels, can greatly impact IDS performance. The consistent benefit of *ALAD* begs the question, what other data or meta-data elements should be considered when constructing HIDS?

The results of our architecture search were fairly inconclusive with respect to classification performance, with WaveNet performing best on ADFA-LD and the LSTM model performing best on PLAID. However, WaveNet required approximately 60% less training time to converge on both ADFA-LD and PLAID when compared with

similarly sized LSTM and GRU models. Thus, practitioners looking to train deep learning empowered HIDS quickly or scale up to massive data sets may prefer architectures composed primarily of convolutions over those composed of recurrent layers.

In our application of allotaxonographs to ADFA-LD and PLAID we identified clear differences between system calls created by baseline and malicious behavior. These differences may lead to additional insights into datasets why deep learning models out perform traditional machine learning models for HIDS. Future work should continue to investigate quantitative methods for corpus divergence in order to improve the interpretability of HIDS.

Overall, our results represent a significant improvement in the state-of-the-art in anomaly-based HIDS. We provide a useful new dataset for the broader HIDS research community, and a blueprint for developing deep learning empowered HIDS by presenting clear evaluation methodologies and reproducible results. Finally, we highlight opportunities for adapting these tools to particular domains.

7.2 Market Inefficencies

We show that total ROC in Russell 3000 securities was well in excess of $2 billion USD during 2016. While consistent with the two comprehensive studies of the modern U.S. market [75, 77], our ROC calculations provide the first empirical evidence explaining how traders might profitably exploit market dislocations, despite paying up to $2.0B USD annually for order flow [111].

Compounding these results, we provide strong statistical evidence that the S&P 500 excluding Dow 30 securities, to which we refer as the SPexDow, is the primary driver

of ROC among the three mutually exclusive categories of equities (Dow 30, SPexDow, and Russell 3000 excluding S&P 500 securities, or the RexSP).

Compounding the above results, we find that structure in the distributions of DS start times and duration persist across the entire Russell 3000, indicating some broader microstructure-based proximate cause of this structure. Distributions of DS duration exhibit a large peak between 10^{-4} and 10^{-3} seconds (100 microseconds to one millisecond), but also exhibit a second smaller, yet distinct, peak near one second. This separation of timescales in the distribution provide evidence for the existence of at least two distinct proximate causes of DS. Distributions of DS start times display even more intricate structure, with large peaks at the beginning and end of the trading day, self-similarity on the half-hour and ten-minute timescales, and a large spike at 2:00 P..

ROC was highest among SPexDow securities, but ROC per share was highest among RexSP securities, which were also the most lightly-traded securities. All time series of ROC exhibit behavior of anti-autocorrelation, meaning that they are mean-reverting. ROC in the SPexDow Granger-cause ROC in the other market categories, but the converse is not true; while the Dow Granger-causes the RexSP, the RexSP only weakly Granger-causes the Dow and does not have any effect on the SPexDow.

Taken together, these results paint the picture of a NMS the physical structure of which generates effects that are persistent across size of equity and exchange. Amplifying these persistent effects is the apparent central role of the SPexDow; in number of DSs, amount of ROC, spectral properties of ROC time series, and Granger-causal relationships, the story emerges of the SPexDow's characteristics being generated by largely-endogenous factors and subsequently influencing the characteristics of the

Dow and RexSP. Future work could explore in more depth the extent to which microstructure effects arising first in the SPexDow then spread to other mutually exclusive market categories and propagate through time. This work could also explore the evolutionary dynamics of the modern NMS from its birth following the financial crisis of 2007/8 to the present day. The NMS may not have remained static, with a constant number of market centers and a stationary distribution of market agents and trading strategies, but rather may have experienced fluctuations in the number of exchanges, in the regulatory environment, and in strategy profiles of trading agents. Such an analysis could pave the way for better informed modelling efforts and the advancement of market theory.

7.3 Application Similarities

Despite the disparate application domains and use cases commonalities were discovered. A discovery of particular interest is the importance of relatively minor meta-data. The inclusion of application-level labels significantly improved the performance of IDSs, and a common stock's inclusion in the S&P 500 appears to have a significant impact on ROC and DSs. This finding underscores the importance of data curation and suggests that the inclusion and creation of meta-data should be carefully considered.

The processes of detecting cyber intrusion and establishing baseline behavior of financial markets are not necessarily all that different. Though they were not considered in this dissertation we note similarity between frequency based approaches to intrusion detection and our DSs analysis. In a similar vein, there exists a class of traders who

make decisions quickly on short sequences of events similar to our *ALAD* pipeline.

Finally, both applications were developed for evolving systems. Developers of IDSs must adapt their systems to emerging attack methods, and a changing computational landscape, such as moving from single to multi-processed applications. Similarly, the NMS is constantly changing, as of writing new exchanges are coming online and sweeping changes to regulations are under consideration.

Bibliography

[1] Martin Joos. The psycho-biology of language, 1936.

[2] Bats. Us equities/options connectivity manual. 2016.

[3] Chicago Board Options Exchange. Us equities/options connectivity manual. 2019.

[4] The Consolidated Tape Association. The consolidated tape association. 2020.

[5] Hongyu Liu and Bo Lang. Machine learning and deep learning methods for intrusion detection systems: A survey. *Applied Sciences*, 9(20):4396, 2019.

[6] LLC. SolarWinds Worldwide. Solarwinds security event manager, 2020. Accessed: 2020-06-16.

[7] Splunk. Splunk intrusion detection system, 2020. Accessed: 2020-06-16.

[8] OSSEC Project Team. Ossec: Host intrusion detection for everyone, 2020. Accessed: 2020-06-16.

[9] Blake E Strom, Andy Applebaum, Doug P Miller, Kathryn C Nickels, Adam G Pennington, and Cody B Thomas. Mitre att&ck: Design and philosophy. *Technical report*, 2018.

[10] R. Cont. Empirical properties of asset returns: stylized facts and statistical issues. *Quantitative Finance*, 1(2):223–236, 2001.

[11] Josef Penso de la Vega. *Confusion Des Confusiones*. Baker Library, Harvard Graduate School of Business Administration, 'S-Gravenhage, 1688.

[12] Louis Bachelier. *ThĂŠorie de la spĂŠculation*. Gauthier-Villars, France, 1900.

[13] Frank Hyneman Knight. Risk, uncertainty and profit. *Houghton Mifflin*, 3, 1921.

[14] Eugene F Fama. The behavior of stock-market prices. *The journal of Business*, 38(1):34–105, 1965.

[15] Andrei Kirilenko, Albert S Kyle, Mehrdad Samadi, and Tugkan Tuzun. The flash crash: The impact of high frequency trading on an electronic market. *Available at SSRN*, 1686004, 2011.

[16] Michael A Goldstein and Kenneth A Kavajecz. Trading strategies during circuit breakers and extreme market movements. *Journal of Financial Markets*, 7(3):301–333, 2004.

[17] Mark Grinblatt and Matti Keloharju. The investment behavior and performance of various investor types: a study of finland's unique data set. *Journal of financial economics*, 55(1):43–67, 2000.

[18] U.S. Securities and Exchange Commission. Strategic plan fiscal years 2014-2018. 2014.

[19] George A. Akerlof. The market for "lemons": Quality uncertainty and the market mechanism. *The Quarterly Journal of Economics*, 84(3):488–500, 1970.

[20] David Easley and Maureen O'hara. Price, trade size, and information in securities markets. *Journal of Financial economics*, 19(1):69–90, 1987.

[21] Alex D Wissner-Gross and Cameron E Freer. Relativistic statistical arbitrage. *Physical Review E*, 82(5):056104, 2010.

[22] Shengwei Ding, John Hanna, and Terrence Hendershott. How slow is the nbbo? a comparison with direct exchange feeds. *Financial Review*, 49(2):313–332, 2014.

[23] Phil Mackintosh. The need for speed. 2014.

[24] Jacob Adrian. Informational inequality: How high frequency traders use premier access to information to prey on institutional investors. *Duke L. & Tech. Rev.*, 14:256, 2016.

[25] John H Ring IV, Colin M Van Oort, Christian Skalka, and Joseph Near. Methods for host-based intrusion detection with deep learning. *Submitted*, 2020.

[26] Brian F Tivnan, David Rushing Dewhurst, Colin M Van Oort, John H Ring IV, Tyler J Gray, Brendan F Tivnan, Matthew TK Koehler, Matthew T McMahon, David M Slater, Jason G Veneman, et al. Fragmentation and inefficiencies in us equity markets: Evidence from the dow 30. *PloS one*, 15(1):e0226968, 2020.

[27] John H. Ring IV au2, Colin M. Van Oort, David R. Dewhurst, Tyler J. Gray, Christopher M. Danforth, and Brian F. Tivnan. Scaling of inefficiencies in the u.s. equity markets: Evidence from three market indices and more than 2900 securities, 2020.

[28] Phil Mackintosh. Time is relativity: What physics has to say about market infrastructure. 2020.

[29] FINRA. Ats transparency data quarterly statistics. https://www.finra.org/industry/otc/ats-transparency-data-quarterly-statistics, Accessed 2019-06-19.

[30] James P Anderson. Computer security threat monitoring and surveillance. *Technical Report, James P. Anderson Company*, 1980.

[31] Erxue Min, Jun Long, Qiang Liu, Jianjing Cui, and Wei Chen. Tr-ids: Anomaly-based intrusion detection through text-convolutional neural network and random forest. *Security and Communication Networks*, 2018, 2018.

[32] Kehe Wu, Zuge Chen, and Wei Li. A novel intrusion detection model for a massive network using convolutional neural networks. *IEEE Access*, 6:50850–50859, 2018.

[33] Yi Zeng, Huaxi Gu, Wenting Wei, and Yantao Guo. $deep-full-range$: A deep learning based network encrypted traffic classification and intrusion detection framework. *IEEE Access*, 7:45182–45190, 2019.

[34] Maria Rigaki and Sebastian Garcia. Bringing a gan to a knife-fight: Adapting malware communication to avoid detection. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 70–75. IEEE, 2018.

[35] Kathleen Goeschel. Reducing false positives in intrusion detection systems using data-mining techniques utilizing support vector machines, decision trees, and naive bayes for off-line analysis. In *SoutheastCon 2016*, pages 1–6. IEEE, 2016.

[36] Phuangpaka Kuttranont, Kobkun Boonprakob, Comdet Phaudphut, Songyut Permpol, Phet Aimtongkhamand, Urachart KoKaew, Boonsup Waikham, and Chakchai So-In. Parallel knn and neighborhood classification implementations on gpu for network intrusion detection. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, 9(2-2):29–33, 2017.

[37] Sasanka Potluri, Shamim Ahmed, and Christian Diedrich. Convolutional neural networks for multi-class intrusion detection system. In *International Conference*

on Mining Intelligence and Knowledge Exploration, pages 225–238. Springer, 2018.

[38] Baoan Zhang, Yanhua Yu, and Jie Li. Network intrusion detection based on stacked sparse autoencoder and binary tree ensemble method. In *2018 IEEE International Conference on Communications Workshops (ICC Workshops)*, pages 1–6. IEEE, 2018.

[39] He Zhang, Xingrui Yu, Peng Ren, Chunbo Luo, and Geyong Min. Deep adversarial learning in intrusion detection: A data augmentation enhanced framework. *arXiv preprint arXiv:1901.07949*, 2019.

[40] Tao Ma, Fen Wang, Jianjun Cheng, Yang Yu, and Xiaoyun Chen. A hybrid spectral clustering and deep neural network ensemble algorithm for intrusion detection in sensor networks. *Sensors*, 16(10):1701, 2016.

[41] Ahmed Ahmim, Leandros Maglaras, Mohamed Amine Ferrag, Makhlouf Derdour, and Helge Janicke. A novel hierarchical intrusion detection system based on decision tree and rules-based models. In *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 228–233. IEEE, 2019.

[42] Xiaoyong Yuan, Chuanhuang Li, and Xiaolin Li. Deepdefense: identifying ddos attack via deep learning. In *2017 IEEE International Conference on Smart Computing (SMARTCOMP)*, pages 1–8. IEEE, 2017.

[43] Benjamin J Radford, Leonardo M Apolonio, Antonio J Trias, and Jim A Simpson. Network traffic anomaly detection using recurrent neural networks. *arXiv preprint arXiv:1803.10769*, 2018.

[44] Wei Wang, Yiqiang Sheng, Jinlin Wang, Xuewen Zeng, Xiaozhou Ye, Yongzhong Huang, and Ming Zhu. Hast-ids: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection. *IEEE Access*, 6:1792–1806, 2017.

[45] Steven McElwee, Jeffrey Heaton, James Fraley, and James Cannady. Deep learning for prioritizing and responding to intrusion detection alerts. In *MILCOM 2017-2017 IEEE Military Communications Conference (MILCOM)*, pages 1–5. IEEE, 2017.

[46] Stephanie Forrest, Steven A Hofmeyr, Anil Somayaji, and Thomas A Longstaff. A sense of self for unix processes. In *Proceedings 1996 IEEE Symposium on Security and Privacy*, pages 120–128. IEEE, 1996.

[47] Massachusetts Institute of Technology Lincoln Laboratory. Darpa intrusion detection evaluation dataset, 1998/1999. https://www.ll.mit.edu/r-d/datasets/1999-darpa-intrusion-detection-evaluation-dataset, https://www.ll.mit.edu/r-d/datasets/1998-darpa-intrusion-detection-evaluation-dat Accessed: 2020-05-12.

[48] ACM Special Interest Group on Knowledge Discovery and Data Mining. Kdd cup 1999: Computer network intrusion detection, 1999. https://www.kdd.org/kdd-cup/view/kdd-cup-1999/Data, http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, Accessed 2020/07/10.

[49] University of New Mexico Computer Science Department. Unm system call dataset, 1998. Accessed: 2020-05-12.

[50] Gideon Creech and Jiankun Hu. Generation of a new ids test dataset: Time to retire the kdd collection. In *2013 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 4487–4492. IEEE, 2013.

[51] John McHugh. Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. *ACM Transactions on Information and System Security (TISSEC)*, 3(4):262–294, 2000.

[52] Ahmad Azab, Robert Layton, Mamoun Alazab, and Jonathan Oliver. Mining malware to detect variants. In *2014 fifth cybercrime and trustworthy computing conference*, pages 44–53. IEEE, 2014.

[53] Ahmad Azab, Mamoun Alazab, and Mahdi Aiash. Machine learning based botnet identification traffic. In *2016 IEEE Trustcom/BigDataSE/ISPA*, pages 1788–1794. IEEE, 2016.

[54] Sitalakshmi Venkatraman, Mamoun Alazab, and R Vinayakumar. A hybrid deep learning image-based analysis for effective malware detection. *Journal of Information Security and Applications*, 47:377–389, 2019.

[55] Weizhi Meng, Wenjuan Li, and Lam-For Kwok. Design of intelligent knn-based alarm filter using knowledge-based alert verification in intrusion detection. *Security and Communication Networks*, 8(18):3883–3895, 2015.

[56] Nam Nhat Tran, Ruhul Sarker, and Jiankun Hu. An approach for host-based intrusion detection system design using convolutional neural network. In *International Conference on Mobile Networks and Management*, pages 116–126. Springer, 2017.

[57] Aaron Tuor, Samuel Kaplan, Brian Hutchinson, Nicole Nichols, and Sean Robinson. Deep learning for unsupervised insider threat detection in structured cybersecurity data streams. In *Workshops at the Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[58] Atul Bohara, Uttam Thakore, and William H Sanders. Intrusion detection in enterprise systems by combining and clustering diverse monitor data. In *Proceedings of the Symposium and Bootcamp on the Science of Security*, pages 7–16, 2016.

[59] Solomon Ogbomon Uwagbole, William J Buchanan, and Lu Fan. Applied machine learning predictive analytics to sql injection attack detection and prevention. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pages 1087–1090. IEEE, 2017.

[60] Ali Moradi Vartouni, Saeed Sedighian Kashi, and Mohammad Teshnehlab. An anomaly detection method to detect web attacks using stacked auto-encoder. In *2018 6th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS)*, pages 131–134. IEEE, 2018.

[61] Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1285–1298, 2017.

[62] Ehsan Aghaei and Gursel Serpen. Ensemble classifier for misuse detection using n-gram feature vectors through operating system call traces. *International Journal of Hybrid Intelligent Systems*, 14(3):141–154, 2017.

[63] Eleazar Eskin, Wenke Lee, and Salvatore J Stolfo. Modeling system calls for intrusion detection with dynamic window sizes. In *Proceedings DARPA Information Survivability Conference and Exposition II. DISCEX'01*, volume 1, pages 165–175. IEEE, 2001.

[64] Andrew P Kosoresow and SA Hofmeyer. Intrusion detection via system call traces. *IEEE software*, 14(5):35–42, 1997.

[65] XA Hoang and Jiankun Hu. An efficient hidden markov model training scheme for anomaly intrusion detection of server applications based on system calls. In *Proceedings. 2004 12th IEEE International Conference on Networks (ICON 2004)(IEEE Cat. No. 04EX955)*, volume 2, pages 470–474. IEEE, 2004.

[66] Eleazar Eskin, Andrew Arnold, Michael Prerau, Leonid Portnoy, and Sal Stolfo. A geometric framework for unsupervised anomaly detection. In *Applications of data mining in computer security*, pages 77–101. Springer, 2002.

[67] Yanxin Wang, Johnny Wong, and Andrew Miner. Anomaly intrusion detection using one class svm. In *Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop, 2004.*, pages 358–364. IEEE, 2004.

[68] Gyuwan Kim, Hayoon Yi, Jangho Lee, Yunheung Paek, and Sungroh Yoon. Lstm-based system-call language modeling and robust ensemble method for designing host-based intrusion detection systems. *arXiv preprint arXiv:1611.01726*, 2016.

[69] ShaoHua Lv, Jian Wang, YinQi Yang, and Jiqiang Liu. Intrusion prediction with system-call sequence-to-sequence model. *IEEE Access*, 6:71413–71421, 2018.

[70] Ashima Chawla, Brian Lee, Sheila Fallon, and Paul Jacob. Host based intrusion detection system with combined cnn/rnn model. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 149–158. Springer, 2018.

[71] Sitalakshmi Venkatraman and Mamoun Alazab. Use of data visualisation for zero-day malware detection. *Security and Communication Networks*, 2018, 2018.

[72] Tran Khanh Dang and Tran Tri Dang. A survey on security visualization techniques for web information systems. *International Journal of Web Information Systems*, 2013.

[73] Peter Sheridan Dodds, Joshua R Minot, Michael V Arnold, Thayer Alshaabi, Jane Lydia Adams, David Rushing Dewhurst, Tyler J Gray, Morgan R Frank, Andrew J Reagan, and Christopher M Danforth. Allotaxonometry and rank-turbulence divergence: A universal instrument for comparing complex systems. *arXiv preprint arXiv:2002.09770*, 2020.

[74] U.S. Securities and Exchange Commission. Staff report on algorithmic trading in u.s. capital markets. 2020.

[75] Elaine Wah. How prevalent and profitable are latency arbitrage opportunities on us stock exchanges? *Available at SSRN 2729109*, 2016.

[76] U.S. Securities and Exchange Commission. Midas: Market information data analytics system. 2013.

[77] Matteo Aquilina, Eric B Budish, and Peter O'Neill. Quantifying the high-frequency trading "arms race": A simple new methodology and estimates. *Chicago Booth Research Paper*, (20-16), 2020.

[78] Maureen O'Hara. High frequency market microstructure. *Journal of Financial Economics*, 116(2):257–270, 2015.

[79] Robert Bloomfield, Maureen O'hara, and Gideon Saar. How noise trading affects markets: An experimental analysis. *The Review of Financial Studies*, 22(6):2275–2302, 2009.

[80] Eric Budish, Peter Cramton, and John Shim. The high-frequency trading arms race: Frequent batch auctions as a market design response. *The Quarterly Journal of Economics*, 130(4):1547–1621, 2015.

[81] Fischer Black. Noise. *The Journal of finance*, 41(3):528–543, 1986.

[82] Eugene F Fama. Efficient capital markets: A review of theory and empirical work. *The Journal of Finance*, 25(2):383–417, 1970.

[83] Jean-Philippe Bouchaud. Econophysics: Still fringe after 30 years? *arXiv preprint arXiv:1901.03691*, 2019.

[84] James Foye, Dusan Mramor, and Marko Pahor. The persistence of pricing inefficiencies in the stock markets of the eastern european eu nations. 2013.

[85] Eugene F Fama and Kenneth R French. Size, value, and momentum in international stock returns. *Journal of financial economics*, 105(3):457–472, 2012.

[86] Neil Johnson, Guannan Zhao, Eric Hunsader, Hong Qi, Nicholas Johnson, Jing Meng, and Brian Tivnan. Abrupt rise of new machine ecology beyond human response time. *Scientific reports*, 3:2627, 2013.

[87] Andrew W Lo. The adaptive markets hypothesis: Market efficiency from an evolutionary perspective. 2004.

[88] Sanford J Grossman and Joseph E Stiglitz. On the impossibility of informationally efficient markets. *The American economic review*, 70(3):393–408, 1980.

[89] Marshall E Blume and Michael A Goldstein. Differences in execution prices among the nyse, the regionals, and the nasd. *Available at SSRN 979072*, 1991.

[90] Charles MC Lee. Market integration and price execution for nyse-listed securities. *The Journal of Finance*, 48(3):1009–1038, 1993.

[91] Joel Hasbrouck. One security, many markets: Determining the contributions to price discovery. *The journal of Finance*, 50(4):1175–1199, 1995.

[92] Michael J Barclay, Terrence Hendershott, and D Timothy McCormick. Competition among trading venues: Information and trading on electronic communications networks. *The Journal of Finance*, 58(6):2637–2665, 2003.

[93] Andriy V Shkilko, Bonnie F Van Ness, and Robert A Van Ness. Locked and crossed markets on nasdaq and the nyse. *Journal of Financial Markets*, 11(3):308–337, 2008.

[94] Jeff Alexander, Linda Giordano, and David Brooks. Dark pool execution quality: A quantitative view. *http://blog.themistrading.com/wp-content/uploads/2015/08/Dark-Pook-Execution-Quality-Short-Final.pdf*, 2015.

[95] James J Angel, Lawrence E Harris, and Chester S Spatt. Equity trading in the 21st century. *The Quarterly Journal of Finance*, 1(01):1–53, 2011.

[96] James J Angel, Lawrence E Harris, and Chester S Spatt. Equity trading in the 21st century: An update. *The Quarterly Journal of Finance*, 5(01):1550002, 2015.

[97] Allen Carrion. Very fast money: High-frequency trading on the nasdaq. *Journal of Financial Markets*, 16(4):680–711, 2013.

[98] Albert J Menkveld. High frequency trading and the new market makers. *Journal of Financial Markets*, 16(4):712–740, 2013.

[99] Michael A Goldstein, Pavitra Kumar, and Frank C Graves. Computerized and high-frequency trading. *Financial Review*, 49(2):177–202, 2014.

[100] Tarun Chordia, Amit Goyal, Bruce N Lehmann, and Gideon Saar. High-frequency trading. 2013.

[101] Sal Arnuk and Joseph Saluzzi. *Broken markets: how high frequency trading and predatory practices on Wall Street are destroying investor confidence and your portfolio*. FT Press, 2012.

[102] U.S. Securities and Exchange Commission. Trade execution. 2013.

[103] U.S. Securities and Exchange Commission. Fast answers: Internalization. 2000.

[104] U.S. Securities and Exchange Commission. Fast answers: Payment for order flow. 2007.

[105] U.S. Securities and Exchange Commission. Citadel securities paying $22 million for misleading clients about pricing trades. 2017.

[106] Dave Michaels. Robinhood settles claims it didn't ensure best prices for customer trades; the online brokerage agreed to pay $1.25 million. *Wall Street Journal*, 2019.

[107] William Power. 'in-house' trades can be costly for small investors. *Wall Street Journal*, page C1, 1994.

[108] Tarun Chordia and Avanidhar Subrahmanyam. Market making, the tick size, and payment-for-order flow: theory and evidence. *Journal of Business*, pages 543–575, 1995.

[109] David Easley, Nicholas M Kiefer, and MAUREEN O'HARA. Cream-skimming or profit-sharing? the curious role of purchased order flow. *The Journal of Finance*, 51(3):811–833, 1996.

[110] U.S. Securities and Exchange Commission. Commission notice: Decimals implementation plan for the equities and options markets. 2020.

[111] Michael Wursthorn and Euirim Choi. Does robinhood make it too easy to trade? from free stocks to confetti; some behavioral researchers say the app's simplicity encourages novice investors to take bigger risks. *Wall Street Journal*, 2020.

[112] Benoit B Mandelbrot. The variation of certain speculative prices. In *Fractals and scaling in finance*, pages 371–418. Springer, 1997.

[113] Benoit B Mandelbrot. *Fractals and scaling in finance: Discontinuity, concentration, risk. Selecta volume E.* Springer Science & Business Media, Chicago, USA, 2013.

[114] H.E. Stanley and V. Plerou. Scaling and universality in economics: empirical results and theoretical interpretation. *Quantitative Finance*, 1(6):563–567, 2001.

[115] Felix Patzelt and Jean-Philippe Bouchaud. Universal scaling and nonlinearity of aggregate price impact in financial markets. *Physical Review E*, 97(1):012304, 2018.

[116] Tiziana Di Matteo. Multi-scaling in finance. *Quantitative finance*, 7(1):21–36, 2007.

[117] Eugene H Stanley, Vasiliki Plerou, and Xavier Gabaix. A statistical physics view of financial fluctuations: Evidence for scaling and universality. *Physica A: Statistical Mechanics and its Applications*, 387(15):3967–3981, 2008.

[118] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[119] Laura Tuttle. Alternative trading systems: Description of ats trading in national market system stocks. 2013.

[120] Hendrik Bessembinder. Price-time priority, order routing, and trade execution costs in nyse-listed stocks. *Order Routing, and Trade Execution Costs in Nyse-Listed Stocks (June 2001)*, 2001.

[121] Nasdaq. Order types and modifiers. 2017.

[122] New York Stock Exchange. Markets: Order types. 2020.

[123] Chicago Board Options Exchange. Order types and routing. 2020.

[124] The Investors Exchange. Order types summary. 2020.

[125] The Consolidated Tape Association. The consolidated tape association. 2020.

[126] The UTP Plan. Unlisted trading priveleges. 2020.

[127] U.S. Securities and Exchange Commission. Regulation national market system. 2005.

[128] U.S. Securities and Exchange Commission. Consolidated audit trail, 2012. https://www.sec.gov/divisions/marketreg/rule613-info.htm, Accessed 2018-02-03.

[129] Goldman Sachs. Sigma x2 form ats-n. https://www.sec.gov/Archives/edgar/data/42352/000095012319009319/xslATS-N_X01/primary_doc.xml, Accessed 2019-11-13.

[130] Kevin Miller. Calculating optical fiber latency. http://www.m2optics.com/blog/bid/70587/Calculating-Optical-Fiber-Latency, Accessed 2017-07-31.

[131] Anova Technologies. Anova technologies network map, 2018. http://anova-tech.com/sample-page/map/, Accessed 2018-07-13.

[132] John H. Ring IV. Uvm ids gitlab repository, 2020. https://gitlab.com/jhring/uvm_threat_stack, Accessed 2020/07/10.

[133] Canonical Ltd. Ubuntu linux, 2018. Accessed: 2020-05-08.

[134] Tatu Ylonen. Ssh–secure login connections over the internet. In *Proceedings of the 6th USENIX Security Symposium*, volume 37, 1996.

[135] Salvatore Sanfilippo. Redis, 2009. Accessed: 2020-05-08.

[136] Robin Verton. cowroot.c, 2016. Accessed: 2020-05-08.

[137] Will Reese. Nginx: the high-performance web server and reverse proxy. *Linux Journal*, 2008(173):2, 2008.

[138] The PHP Group. PHP hypertext processor, 2016. Accessed: 2020-05-08.

[139] Netcraft Ltd. April 2020 web server survey, 2020. Accessed: 2020-05-08.

[140] OffSec Services Limited. Kali linux, 2019. Accessed: 2020-05-08.

[141] Oracle. Virtual box, 2019. Accessed: 2020-05-08.

[142] Metasploit. Redis attack, 2019. Accessed: 2020-06-02.

[143] Emil Lerner. Php-fpm attack, 2019. Accessed: 2020-06-02.

[144] Robin Verton. Privilege escalation attack, 2019. Accessed: 2020-06-02.

[145] OffSec Services Limited. Brute-force password attack, 2013. Accessed: 2020-06-02.

[146] Steve Klabnik and Carol Nichols. *The Rust Programming Language (Covers Rust 2018)*. No Starch Press, 2019.

[147] S&P Dow Jones Indices. Dow jones averages methodology. 2020.

[148] S&P Dow Jones Indices. S&p u.s. indices methodology. 2020.

[149] FTSE Russell. Russell u.s. equity indexes, construction and methodology. 2020.

[150] Thesys Technologies. Thesys technologies. 2020.

[151] Nathaniel Popper and Ben Protess. To regulate rapid traders, s.e.c. turns to one of them, 2012. https://www.nytimes.com/2012/10/08/business/sec-regulators-turn-to-high-speed-trading-firm.html, Accessed 2018-04-21.

[152] FTSE Russell. Russell 3000 fact sheet. 2020.

[153] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[154] Seth Lloyd. Computational capacity of the universe. *Physical Review Letters*, 88(23):237901, 2002.

[155] Keras Team. Keras tuner, 2020. Accessed: 2020-05-10.

[156] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[157] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318, 2013.

[158] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1:206–215, 05 2019.

[159] Massimo Bernaschi, Emanuele Gabrielli, and Luigi V Mancini. Operating system enhancements to prevent the misuse of system calls. In *Proceedings of the 7th ACM conference on Computer and communications security*, pages 174–183, 2000.

[160] Sal Arnuk and Joseph Saluzzi. Latency arbitrage: The real power behind predatory high frequency trading. 2009.

[161] Robert A Jarrow and Philip Protter. A dysfunctional role of high frequency trading in electronic markets. *International Journal of Theoretical and Applied Finance*, 15(03):1250022, 2012.

[162] Joel Hasbrouck and Gideon Saar. Low-latency trading. *Journal of Financial Markets*, 16(4):646–679, 2013.

[163] Clive WJ Granger. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: Journal of the Econometric Society*, pages 424–438, 1969.

[164] John H. Ring IV. Uvm compfi lab home page. 2020.

[165] C-K Peng, Sergey V Buldyrev, Shlomo Havlin, Michael Simons, H Eugene Stanley, and Ary L Goldberger. Mosaic organization of dna nucleotides. *Physical review e*, 49(2):1685, 1994.

[166] C Bonferroni. Teoria statistica delle classi e calcolo delle probabilita. *Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commericiali di Firenze*, 8:3–62, 1936.

Appendix A

Appendix

A.1 Allotaxonographs

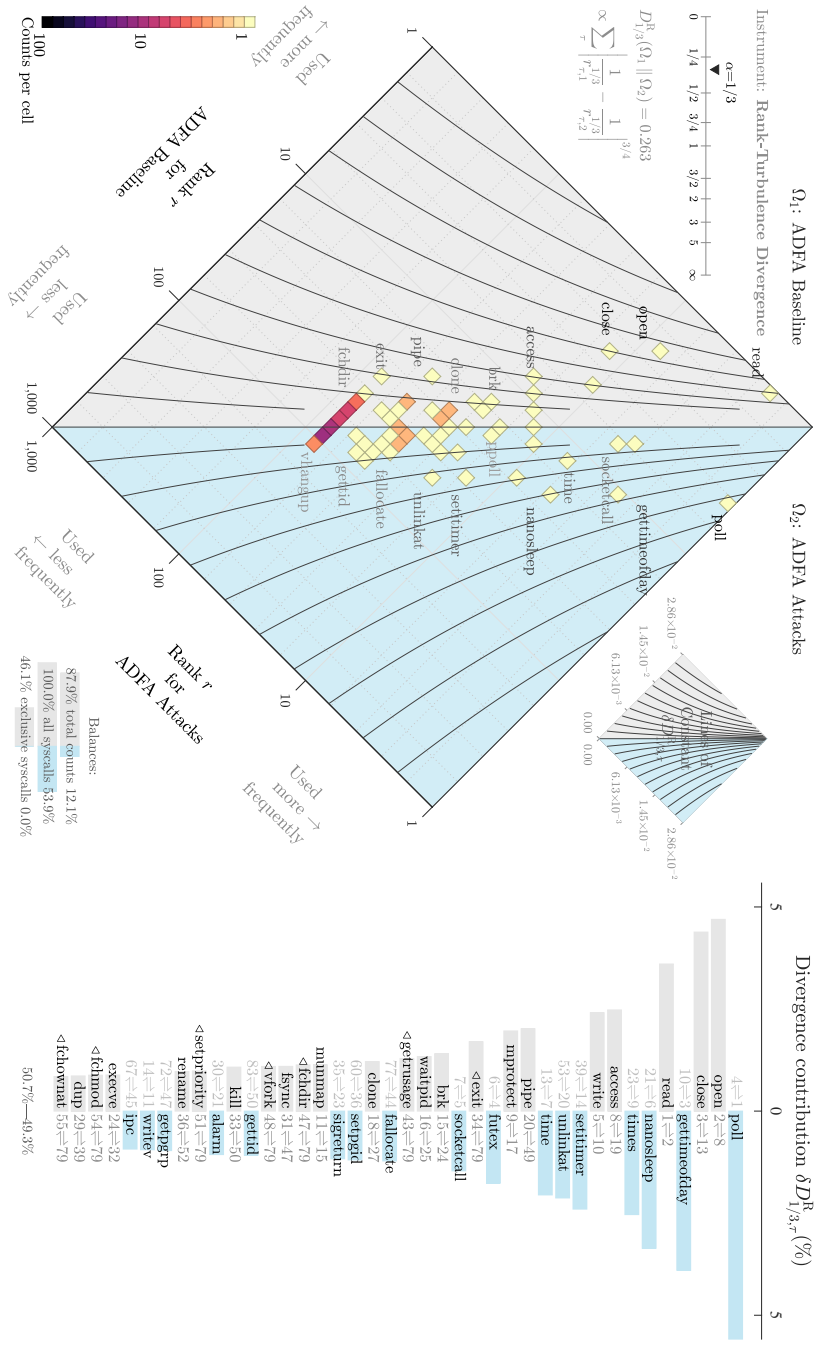


Figure A.1: Comparison of system call rankings between attack and baseline traces in ADFA-LD. Note that some of the most frequently utilized system calls, `poll` *and* `read`*, are among the largest contributors to divergence. Of additional interest is that the most dangerous system calls are not top contributors to divergence.*

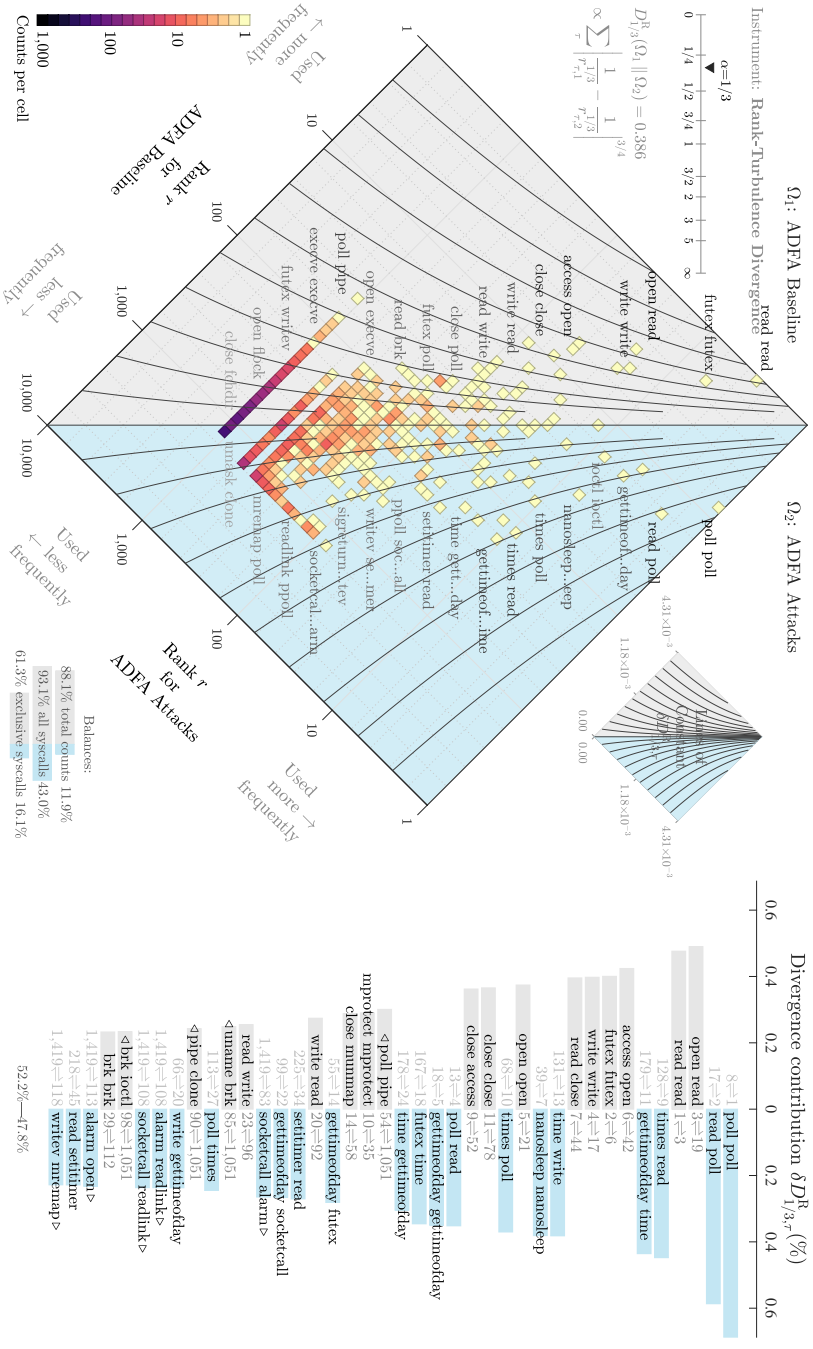


Figure A.2: Comparison of system call bi-gram rankings between attack and baseline traces in ADFA-LD. Similar to uni-grams frequent bi-grams remain top contributors to divergence. We see a larger portion of bi-grams appearing only in one split compared to uni-grams.

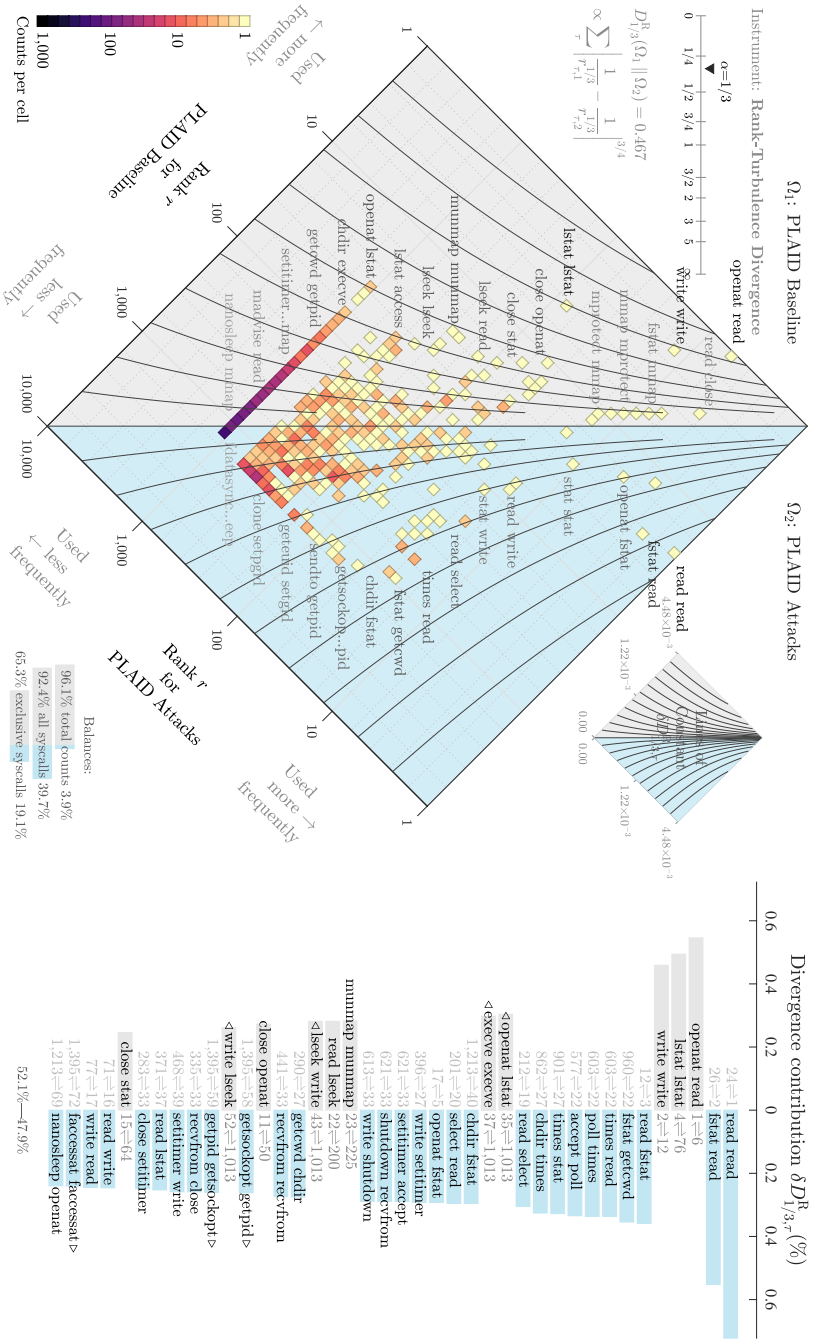


Figure A.3: Comparison of system call bi-gram rankings between attack and baseline traces in PLAID. Similar to uni-grams frequent bi-grams remain top contributors to divergence. We see a larger portion of bi-grams appearing only in one split compared to uni-grams.

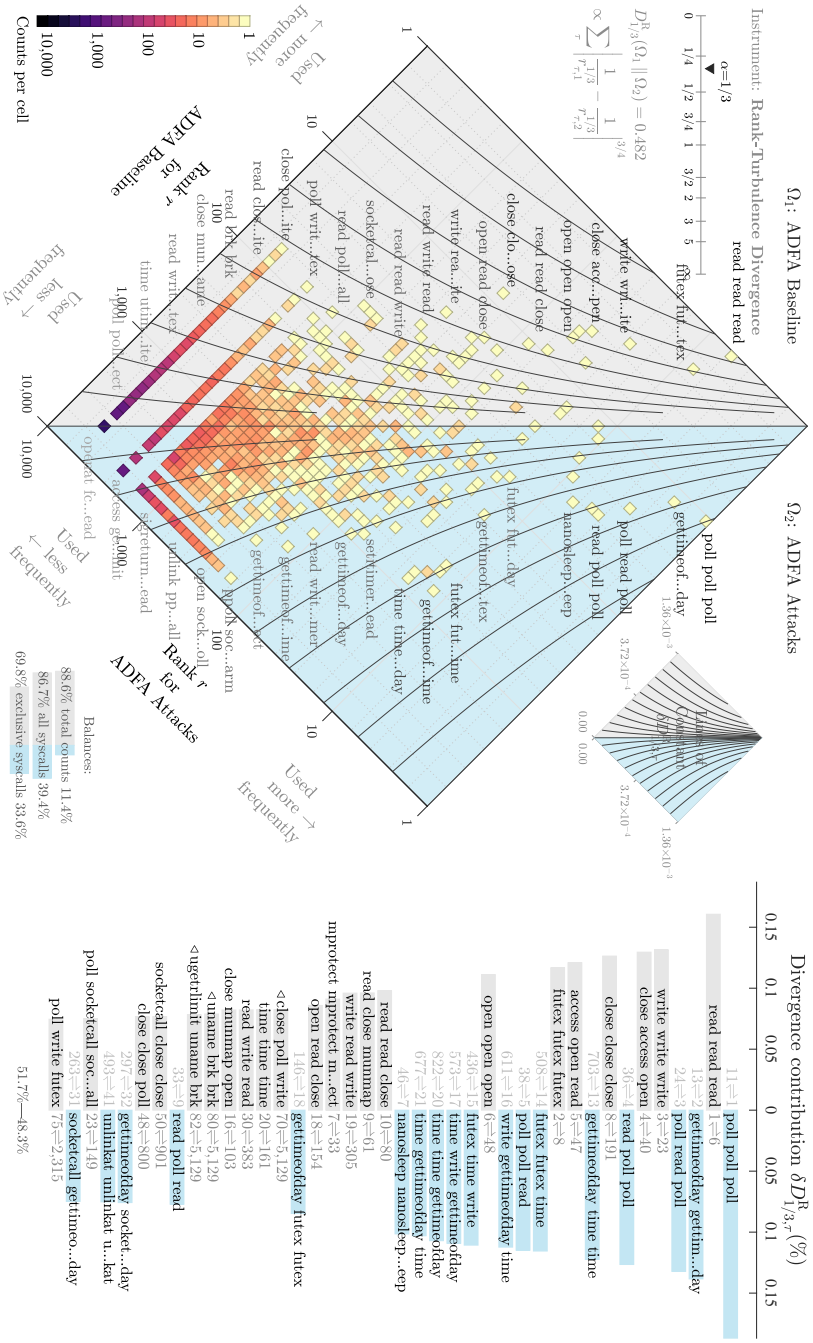


Figure A.4: Comparison of system call tri-gram rankings between attack and baseline traces in ADFA-LD. A slightly larger portion of tri-grams are present only in one set compared to bi-grams. This suggests that longer n-grams help to differentiate between sets.

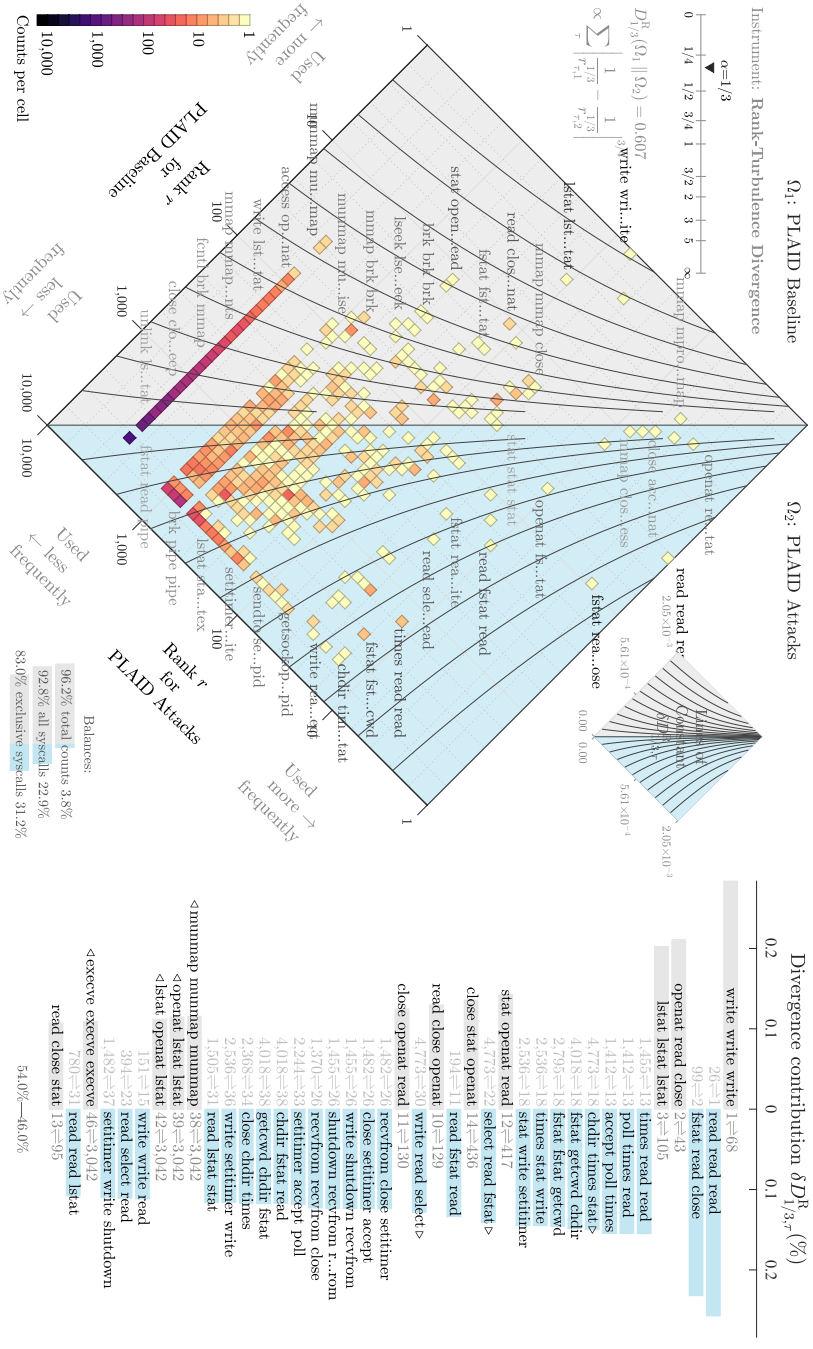


Figure A.5: Comparison of system call tri-gram rankings between attack and baseline traces in PLAID. A slightly larger portion of tri-grams are present only in one set compared to bi-grams. This suggests that longer n-grams help to differentiate between sets.

A.2 System Call Frequencies

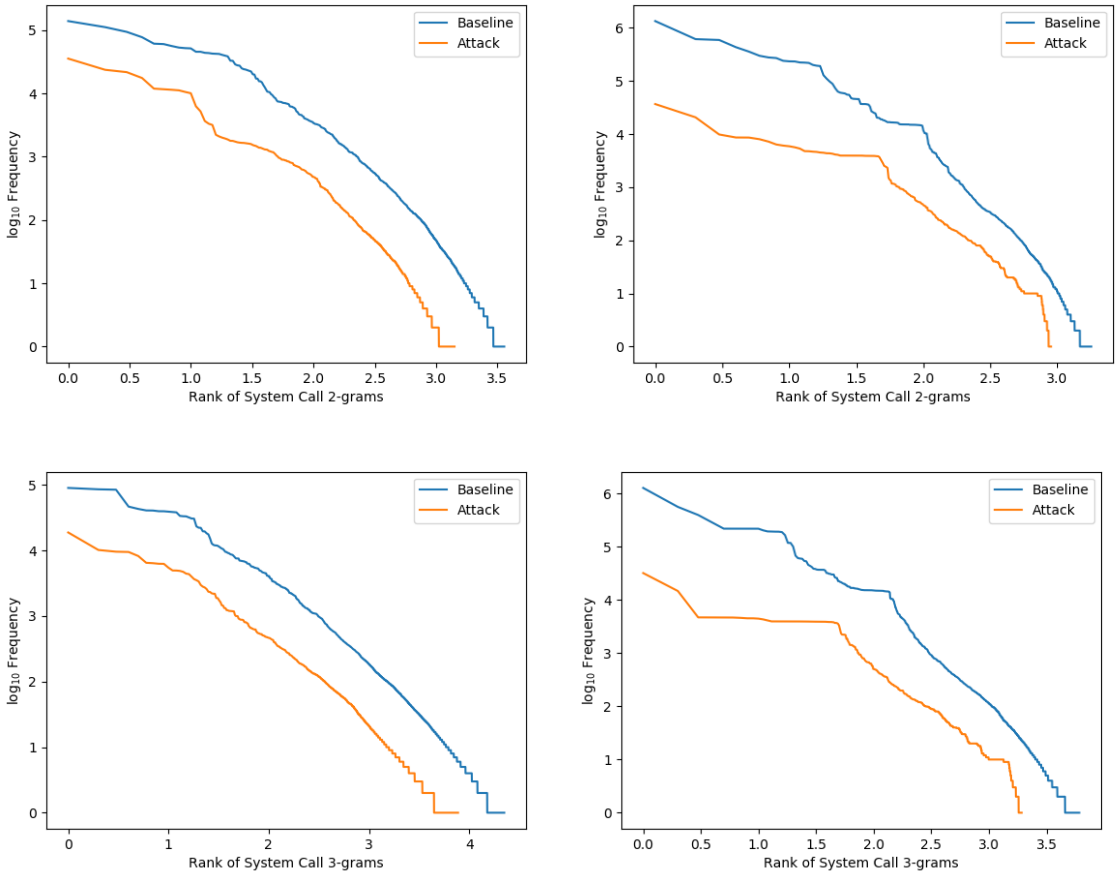


Figure A.6: Rank frequency plots of system call bi (top) abd tri (bottom) grams for attack and baseline traces in ADFA-LD (left) and PLAID (right). The rank frequency appears to approximate a power-law with an exponential cutoff in the tail. Natural language corpora tend to be and stay power-law like for uni through tri-grams with the tail starting to flatten. In contrast to system call corpora which become more power law like.

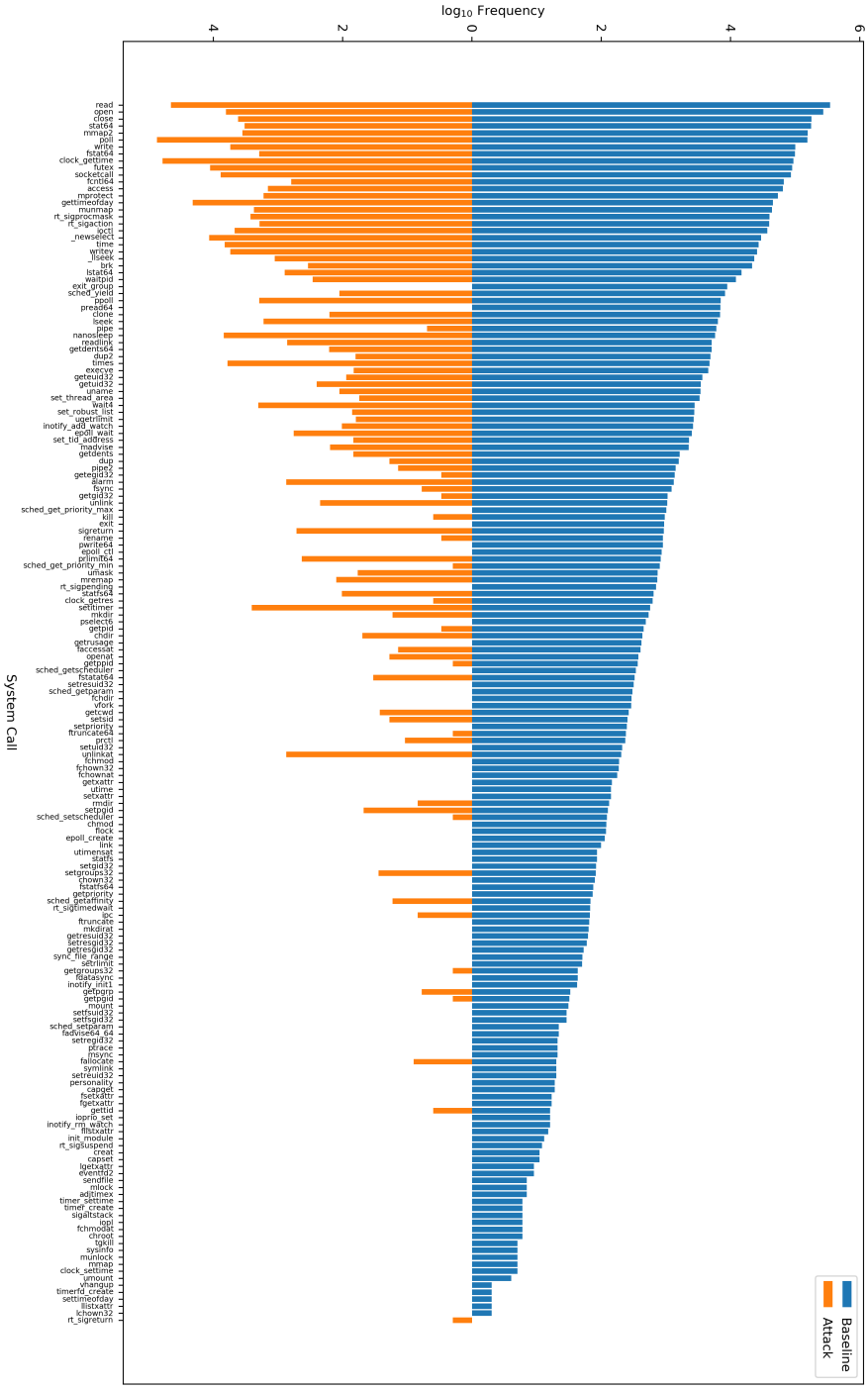


Figure A.7: Comparison of system call usage between baseline and attack traces in ADFA-LD. System calls are in monotonically non-increasing order base on their frequency in baseline traces. Notice that usages of individual system calls differ significantly between sets.

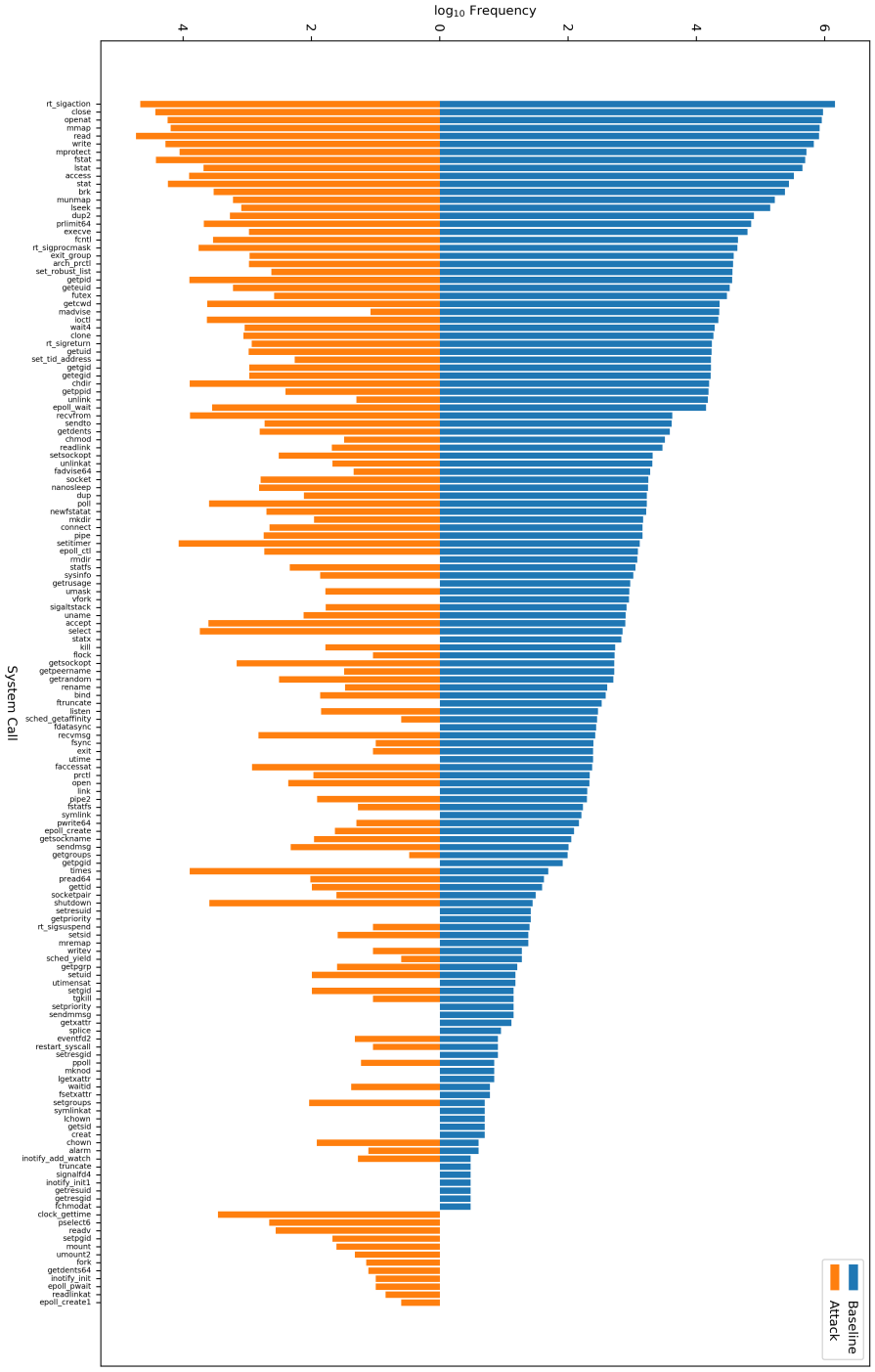


Figure A.8: Comparison of system call usage between baseline and attack traces in PLAID. System calls are in monotonically non-increasing order base on their frequency in baseline traces. Notice that usages of individual system calls differ significantly between sets. Of additional interest is the amount of `clock_gettime` *calls in the attack split.*

A.3 NMS Tables

Russ 3K′	1	Realized Opportunity Cost	$2,013,458,668.87
	2	SIP Opportunity Cost	$1,876,048,519.06
	3	Direct Opportunity Cost	$137,410,149.76
	4	Trades	4,658,307,833
	5	Diff. Trades	1,105,201,803
	6	Traded Value	$24,352,760,600,270.47
	7	Diff. Traded Value	$6,272,439,590,589.91
	8	Percent Diff. Trades	23.73
	9	Percent Diff. Traded Value	25.76
	10	Ratio of 9 / 8	1.0855
RexSP	1	Realized Opportunity Cost	$948,743,328.62
	2	SIP Opportunity Cost	$911,950,130.85
	3	Direct Opportunity Cost	$36,793,197.77
	4	Trades	2,093,415,072
	5	Diff. Trades	482,055,297
	6	Traded Value	$6,669,357,410,332.23
	7	Diff. Traded Value	$1,705,272,719,045.67
	8	Percent Diff. Trades	23.03
	9	Percent Diff. Traded Value	25.57
	10	Ratio of 9 / 8	1.1104
S&P 500′	1	Realized Opportunity Cost	$1,064,715,340.25
	2	SIP Opportunity Cost	$964,098,388.26
	3	Direct Opportunity Cost	$100,616,951.99
	4	Trades	2,564,892,761
	5	Diff. Trades	623,146,506
	6	Traded Value	$18,429,250,470,003.83
	7	Diff. Traded Value	$4,567,166,871,544.24
	8	Percent Diff. Trades	24.30
	9	Percent Diff. Traded Value	25.83
	10	Ratio of 9 / 8	1.0631
	1	Realized Opportunity Cost	$904,501,417.30
	2	SIP Opportunity Cost	$842,017,261.86
	3	Direct Opportunity Cost	$62,484,155.44

	4	Trades	2,172,791,182
	5	Diff. Trades	535,714,275
	6	Traded Value	$13,824,440,155,934.76
	7	Diff. Traded Value	$3,666,630,946,582.52
	8	Percent Diff. Trades	24.66
	9	Percent Diff. Traded Value	26.52
	10	Ratio of 9 / 8	1.0757
Dow 30′	1	Realized Opportunity Cost	$160,213,922.95
	2	SIP Opportunity Cost	$122,081,126.40
	3	Direct Opportunity Cost	$38,132,796.55
	4	Trades	392,101,579
	5	Diff. Trades	87,432,231
	6	Traded Value	$3,858,963,034,003.48
	7	Diff. Traded Value	$900,535,924,961.72
	8	Percent Diff. Trades	22.30
	9	Percent Diff. Traded Value	23.34
	10	Ratio of 9 / 8	1.0465

Table A.5: Summary statistics of realized opportunity cost (ROC) for various equity groups under study during 2016.

Conditioned		min magnitude ($)	max magnitude ($)	duration (s)
None	count	4,011,848.7333		
	mean	0.0110	0.0136	0.075413
	std	0.0391	0.2725	5.829465
	min	0.0100	0.0100	<0.000001
	25%	0.0100	0.0100	0.000248
	50%	0.0100	0.0100	0.000669
	75%	0.0100	0.0103	0.001253
	max	44.6933	279.2057	8,408.931478
Duration	count	2,169,106.5333		
	mean	0.0108	0.0149	0.132779
	std	0.0436	0.3548	7.645375
	min	0.0100	0.0100	0.000546
	25%	0.0100	0.0100	0.000783
	50%	0.0100	0.0100	0.001129
	75%	0.0100	0.0107	0.002654
	max	43.4150	279.1987	8,408.931478
Duration & Magnitude	count	95,757.8000		
	mean	0.0427	0.2370	0.955731
	std	0.3355	1.6130	48.214785
	min	0.0200	0.0200	0.000546
	25%	0.0200	0.0200	0.000698
	50%	0.0200	0.0227	0.001073
	75%	0.0307	0.0433	0.003552
	max	43.4150	114.3480	7,186.866464

Table A.1: Mean of dislocation segment summary statistics taken across the 30 members of the Dow. $545\mu s$ *is used for duration conditioning and* \$0.01 *is used for magnitude conditioning.*

Conditioned		min magnitude ($)	max magnitude ($)	duration (s)
None	count	2,525,082.0448		
	mean	0.0135	0.0168	0.252981
	std	0.2801	0.3996	9.325161
	min	0.0100	0.0100	<0.000001
	25%	0.0100	0.0100	0.000227
	50%	0.0100	0.0101	0.000583
	75%	0.0115	0.0136	0.001085
	max	476.1177	522.6072	9,084.040084
Duration	count	1,189,460.6682		
	mean	0.0134	0.0185	0.555820
	std	0.4601	0.6076	13.029491
	min	0.0100	0.0100	0.000546
	25%	0.0100	0.0100	0.000754
	50%	0.0102	0.0107	0.001119
	75%	0.0117	0.0160	0.008169
	max	471.7331	515.4222	9,084.040084
Duration & Magnitude	count	114,770.0224		
	mean	0.0557	0.1249	1.591543
	std	1.9177	2.5050	54.064998
	min	0.0200	0.0200	0.000546
	25%	0.0202	0.0209	0.000717
	50%	0.0228	0.0346	0.001240
	75%	0.0375	0.0625	0.027820
	max	471.7331	506.9715	6,943.106256

Table A.2: Mean of dislocation segment summary statistics taken across 446 members of the SPexDow. $545\mu s$ *is used for duration conditioning and* $0.01 *is used for magnitude conditioning.*

Conditioned		min magnitude ($)	max magnitude ($)	duration (s)
None	count	770,577.8246		
	mean	0.9734	1.1361	4.413179
	std	34.0534	37.7472	50.079342
	min	0.0100	0.0100	<0.000001
	25%	0.0116	0.0121	0.000245
	50%	0.0139	0.0149	0.001042
	75%	0.0225	0.0302	0.013774
	max	2,238.1205	2,514.9617	8,796.956807
Duration	count	287,399.7217		
	mean	1.2116	1.7162	12.749530
	std	37.6277	46.3599	83.465004
	min	0.0100	0.0100	0.000546
	25%	0.0110	0.0118	0.002065
	50%	0.0147	0.0188	0.072213
	75%	0.0263	0.0408	0.975526
	max	2,033.1633	2,302.4541	8,796.956807
Duration & Magnitude	count	45,062.3366		
	mean	2.1734	3.0486	13.154607
	std	53.2211	66.0958	112.101259
	min	0.0200	0.0200	0.000546
	25%	0.0239	0.0272	0.003933
	50%	0.0338	0.0449	0.053583
	75%	0.0611	0.0806	0.798791
	max	2,033.9931	2,295.6782	7,139.075345

Table A.3: Mean of dislocation segment summary statistics taken across the 2451 members of the RexSP. $545\mu s$ *is used for duration conditioning and* $0.01 *is used for magnitude conditioning.*

Conditioned		min magnitude ($)	max magnitude ($)	duration (s)
None	count	6,431,595.4444		
	mean	0.0216	0.0273	0.339145
	std	0.0856	0.1027	13.327128
	min	0.0100	0.0100	<0.000001
	25%	0.0100	0.0100	0.000284
	50%	0.0100	0.0100	0.000602
	75%	0.0122	0.0156	0.001175
	max	9.0956	9.3744	5,658.596041
Duration	count	3,674,884.7778		
	mean	0.0223	0.0289	0.683211
	std	0.0859	0.1077	18.991011
	min	0.0100	0.0100	0.000546
	25%	0.0100	0.0100	0.000726
	50%	0.0100	0.0111	0.001064
	75%	0.0122	0.0167	0.002494
	max	6.3278	8.4556	5,658.596041
Duration & Magnitude	count	130,853.7778		
	mean	0.1707	0.1800	0.933693
	std	0.2804	0.2995	26.558084
	min	0.0200	0.0200	0.000546
	25%	0.0200	0.0200	0.000765
	50%	0.0344	0.0411	0.001213
	75%	0.1733	0.2933	0.005725
	max	6.3278	8.4311	5,005.870452

Table A.4: Mean of dislocation segment summary statistics taken across the 9 ETFs under study. $545\mu s$ *is used for duration conditioning and* $0.01 *is used for magnitude conditioning.*

	Trades	Traded Val. ($)	Diff. Trades	Diff. Traded Val. ($)	ROC ($)	ROC/Share
count	720,991	720,991	720,991	720,991	720,991	720,991
mean	6,460.98	33,776,788.61	1,532.89	8,699,747.42	2,792.63	0.020880
std	13,249.67	109,021,779.70	3,036.98	25,738,960.57	17,611.14	0.087810
min	0	0	0	0	0	0
25%	599	1,118,022.02	101	199,882.83	237.6100	0.009510
50%	2,020	5,316,322.22	450	1,246,241.41	826.6000	0.011448
75%	6,478	24,797,793.44	1,600	6,525,124.17	2,578.75	0.018836
max	517,270	8,280,915,338.59	103,885	1,596,912,962.05	6,798,041.07	19.3381

Table A.6: Purse statistics for all stocks under study in 2016. The data used to construct this table is aggregated by date and stock, resulting in 720,991 data points that correspond with the 731,556 combinations of 252 trading days in 2016 and 2903 stocks under study.

		Trades	Traded Val. ($)	Diff. Trades	Diff. Traded Val. ($)	ROC ($)	ROC/Share
Russ 3K′	mean	18,485,348.54	96,637,938,889.96	4,385,721.44	24,890,633,295.99	7,989,915.35	0.023073
	std	3,705,825.95	17,507,577,514.36	1,222,558.47	5,929,581,247.64	2,363,234.20	0.003143
	min	7,045,815	41,324,500,835.46	1,197,040	8,277,978,080.59	2,717,631.16	0.018414
	25%	16,178,390	85,348,849,125.71	3,674,541	21,481,677,427.57	6,560,601.80	0.020888
	50%	17,837,416.50	94,176,286,443.74	4,257,438.50	24,165,074,815.55	7,524,560.38	0.022379
	75%	20,114,165.50	103,932,196,142.46	4,964,932.50	27,054,706,014.87	8,884,110.40	0.024693
	max	32,913,872	169,395,493,215.29	9,253,338	47,500,228,278.03	19,622,594.00	0.051371
RexSP	mean	8,307,202.67	26,465,704,009.25	1,912,917.85	6,766,955,234.31	3,764,854.48	0.022109
	std	1,370,512.88	3,786,979,882.64	473,884.96	1,299,054,438.46	1,048,372.83	0.002874
	min	3,183,224	11,363,776,182.38	487,500	2,268,729,995.29	1,436,093.46	0.017744
	25%	7,528,810.25	24,222,297,224.76	1,648,499.25	6,053,458,251.52	3,182,173.91	0.020092
	50%	8,175,352.50	26,166,834,634.22	1,921,121.50	6,779,433,456.68	3,564,482.05	0.021393
	75%	9,061,096.50	28,685,877,060.20	2,161,350.50	7,599,965,429.85	4,206,538.80	0.023737
	max	13,408,508	41,337,807,991.92	3,537,890	10,627,257,029.61	10,083,342.57	0.047415
S&P 500′	mean	10,178,145.88	70,172,234,880.71	2,472,803.60	18,123,678,061.68	4,225,060.87	0.014624
	std	2,406,751.15	14,303,150,882.94	775,201.38	4,760,162,875.50	1,531,548.30	0.002019
	min	3,862,591	29,960,724,653.08	709,540	5,941,906,620.96	1,281,537.70	0.011127
	25%	8,716,552.50	60,764,387,798.11	2,034,844.50	15,251,685,767.67	3,371,948.52	0.013502
	50%	9,684,039	67,776,548,100.32	2,310,806	17,479,288,594.91	3,918,496.70	0.014407
	75%	11,120,226.50	75,672,607,052.02	2,783,838.50	20,074,235,595.26	4,654,693.39	0.015434
	max	19,505,364	128,057,685,223.37	5,715,448	37,114,729,300.67	14,335,072.09	0.031484
SPexDow	mean	8,622,187.23	54,858,889,507.68	2,125,850.30	14,550,122,803.90	3,589,291.34	0.014818
	std	1,960,102.37	10,686,728,768.81	632,025.23	3,571,347,460.11	1,119,395.15	0.002029
	min	3,283,385	23,296,053,599.93	619,976	4,906,051,591.25	1,136,332.05	0.011271
	25%	7,398,970.25	48,123,050,130.46	1,762,152.75	12,329,749,894.94	2,915,802.29	0.013729
	50%	8,237,387.50	53,383,376,977.72	2,006,091.50	14,073,439,429.50	3,384,654.11	0.014579
	75%	9,405,905.75	59,188,646,444.18	2,398,085.25	15,973,362,072.81	4,050,343.31	0.015660
	max	15,909,358	99,048,039,796.82	4,642,419	27,685,776,913.57	9,097,891.31	0.032760
Dow 30′	mean	1,555,958.65	15,313,345,373.03	346,953.30	3,573,555,257.78	635,769.54	0.011792
	std	463,558.93	3,891,299,900.31	146,677.85	1,234,882,079.43	655,911.15	0.008071
	min	579,206	6,664,671,053.15	89,564	1,035,855,029.71	145,205.65	0.008879
	25%	1,278,813.25	12,915,031,172.08	262,209	2,804,569,367.64	417,485.73	0.009667
	50%	1,429,062	14,431,597,662.01	309,158	3,274,390,601.60	514,856.64	0.010213
	75%	1,715,351.25	16,829,521,684.38	387,772	3,993,470,514.97	666,268.27	0.011288
	max	3,596,006	30,999,914,293.66	1,073,029	9,428,952,387.10	7,817,684.58	0.093108

Table A.7: Aggregated purse statistics for different groups of securities in 2016. Each section is composed of date aggregated data, resulting in 252 data points that correspond with the 252 trading days in 2016.

	Skew	Kurtosis
Dow	52.59	3122.65
SPexDow	55.66	5644.74
RexSP	300.12	110365.89

Table A.8: Skew and kurtosis for daily ROC by mutually-exclusive market category, highlighting the remarkably heavy-tailed nature of these distributions.

1	Realized Opportunity Cost	$38,458,070.79
2	SIP Opportunity Cost	$37,970,135.30
3	Direct Opportunity Cost	$487,935.49
4	Trades	86,725,286
5	Diff. Trades	19,612,214
6	Traded Value	$3,678,242,397,422.43
7	Diff. Traded Value	$804,917,872,051.93
8	Percent Diff. Trades	22.61
9	Percent Diff. Traded Value	21.88
10	Ratio of 9 / 8	0.9677

Table A.9: Summary statistics for realized opportunity cost (ROC) observed in the ETFs under study. It is notable that, of all market subsets we study, only this small subset has a ratio of the fraction of differing traded value to fraction of differing trades with value below unity. On a per-trade basis, this means that there is on average less potential for ROC.

	Trades	Traded Value ($)	Diff. Trades	Diff. Traded Value ($)	ROC ($)	ROC/Share
mean	38,391.01	1,628,261,353.44	8,681.81	356,316,012.42	17,024.38	0.021169
std	106,302.46	4,663,474,508.49	23,900.69	1,033,570,406.20	48,481.79	0.043449
min	1	72.4600	0	0	0	0
25%	14	262,574.18	3	48,125.50	35.0000	0.008350
50%	683	15,165,081.37	181	3,386,159.33	455.2200	0.009997
75%	12,121.50	283,540,074.38	4,136	93,960,790.38	6,033.43	0.014408
max	974,888	40,617,035,891.21	251,657	11,028,368,359.92	499,906.77	1.0200

Table A.10: Aggregated purse statistics for the ETFs under study. The data used to construct this table is aggregated by date and instrument, resulting in 2,259 data points that correspond with the 2,268 combinations of 252 trading days in 2016 and 9 ETFs under study.

	Trades	Traded Value ($)	Diff. Trades	Diff. Traded Value ($)	ROC ($)	ROC/Share
mean	344,147.96	14,596,199,989.77	77,826.25	3,194,118,539.89	152,611.39	0.189762
std	157,107.76	6,043,079,696.41	45,179.00	1,675,731,349.39	85,509.19	0.118446
min	113,860	5,018,912,183.01	14,610	703,559,994.91	30,989.52	0.054358
25%	237,021.25	10,471,387,904.01	47,237.50	2,052,459,478.17	94,488.20	0.106098
50%	308,705	13,005,695,875.47	66,509	2,780,132,908	131,084.42	0.169572
75%	394,822.25	16,641,275,220.96	94,108	3,799,483,257.76	186,174.78	0.256871
max	1,177,148	44,900,644,748.00	339,480	12,945,336,256.63	616,859.86	1.0963

Table A.11: Aggregated purse statistics for the ETFs under study. The data used to construct this table is aggregated by date, resulting in 252 data points that correspond with the 252 trading days in 2016.

A.4 NMS Figures

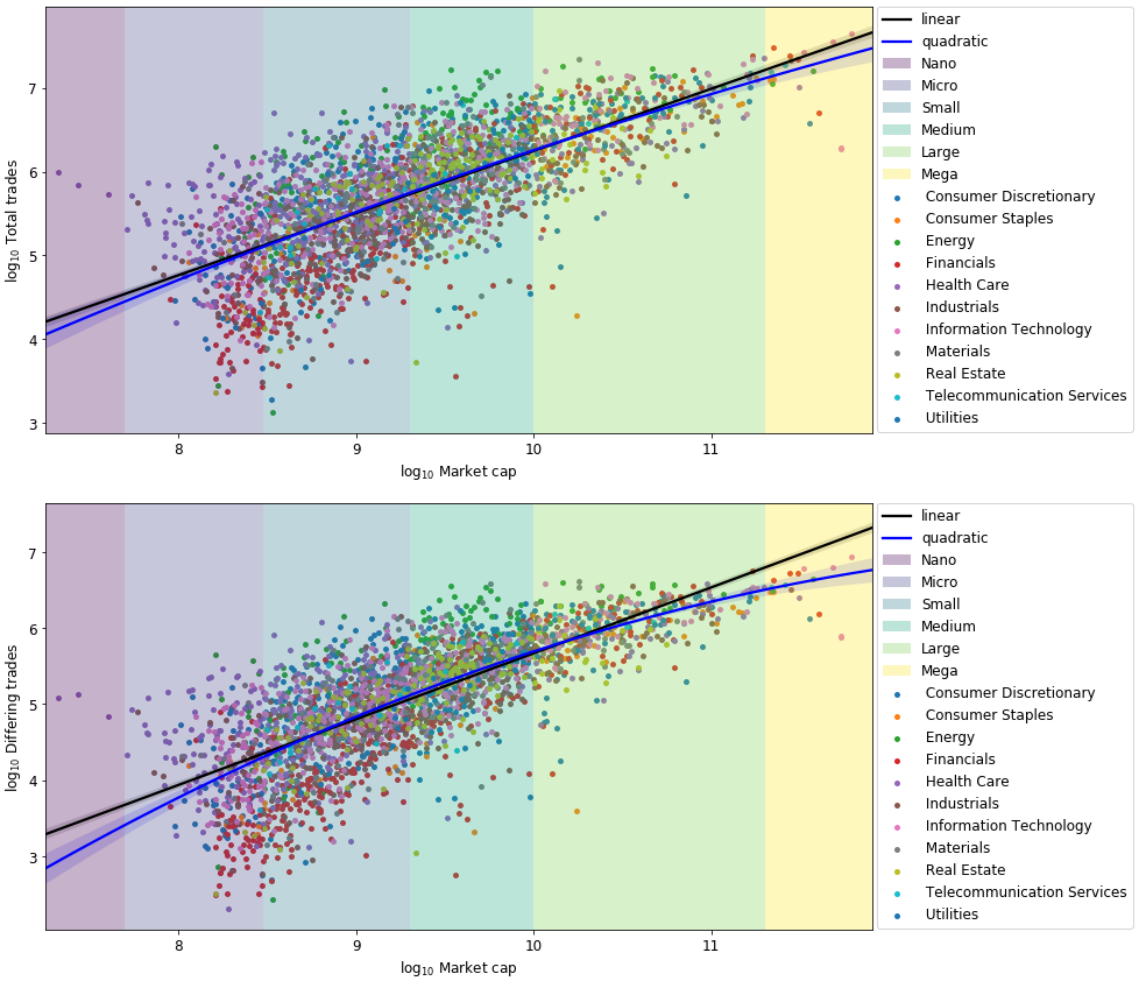


Figure A.9: Relationships between Market Capitalization (MC) and total trades (top) or differing trades (bottom). Similar to Figure **??**, *there is a strong positive relationship in both regressions, along with the same nonlinearity and heteroskedasticity. The data are well-fit by linear and quadratic functions in doubly-logarithmic space. The shaded area surrounding the regression curves indicate 95% confidence intervals for the true curves, calculated using bootstrapping techniques.*

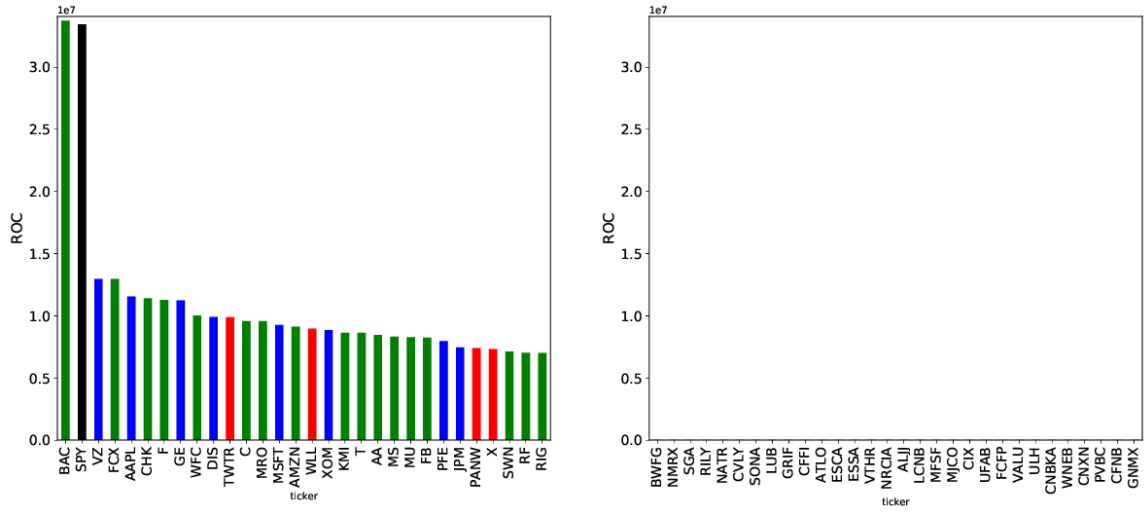


Figure A.10: ROC by ticker ($) for the top 30 (left panel) and bottom 30 (right panel) of all securities under study, ranked by ROC. Constituents of the Dow 30 are shown in blue, constituents of the S&P 500 (excluding the Dow 30) are shown in green, constituents of the Russell 3000 (excluding the S&P 500) are shown in red, and ETFs are shown in black.

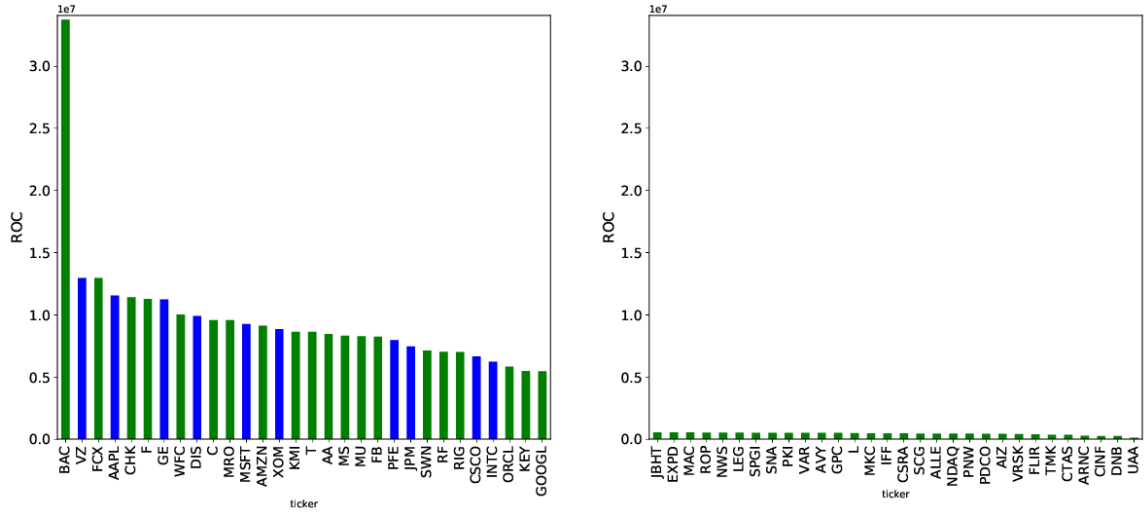


Figure A.11: ROC by ticker ($) for the top 30 (left panel) and bottom 30 (right panel) of S&P 500 securities, ranked by ROC. Constituents of the Dow 30 are shown in blue, while those belonging to the S&P 500 (excluding the Dow 30) are shown in green.

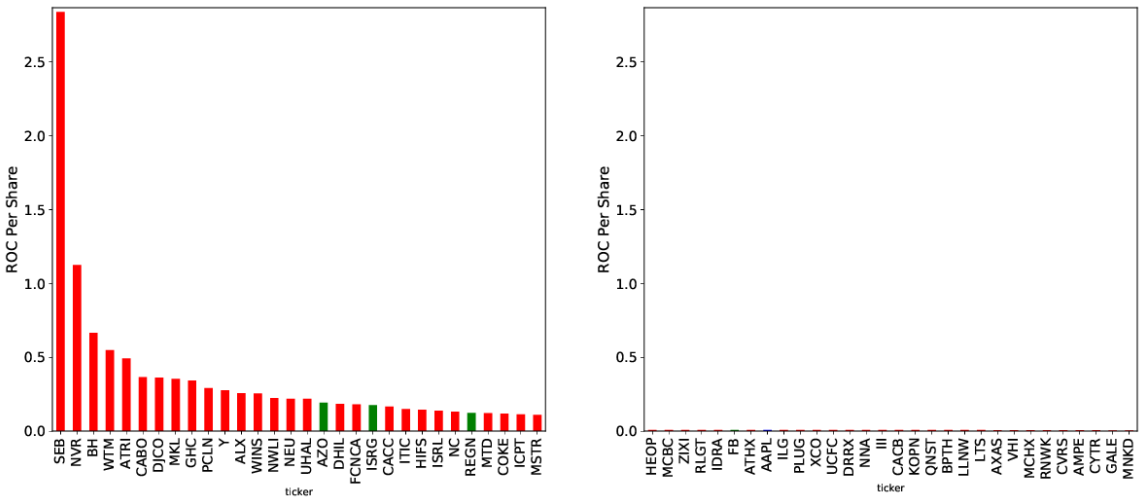


Figure A.12: ROC per share ($ / share) by ticker for the top 30 (left panel) and bottom 30 (right panel) of all securities under study, ranked by ROC. Constituents of the Dow 30 are shown in blue, constituents of the S&P 500 (excluding the Dow 30) are shown in green, and constituents of the Russell 3000 (excluding the S&P 500) are shown in red.

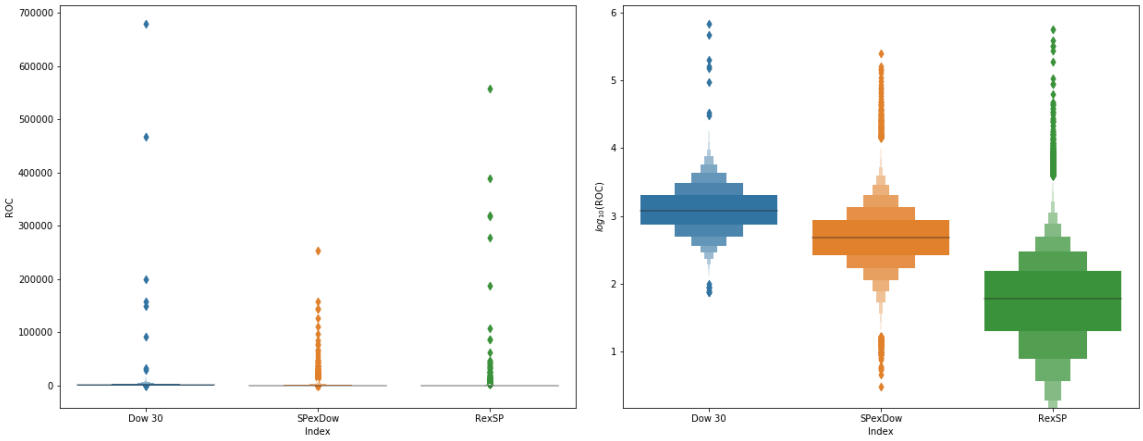


Figure A.13: Distributions of mean ROC per day over the members of each mutually exclusive market category. Linear (left) and log 10 (right) vertical axis scaling are used to provide additional perspective. On average, members of the Dow experience more ROC than members of the SPexDow, which experience more ROC than the RexSP. These distributions are extremely heavy tailed, thus the use of log scaling, and feature a high degree of overlap. Thus there are members from each category that experience high ROC and low ROC.

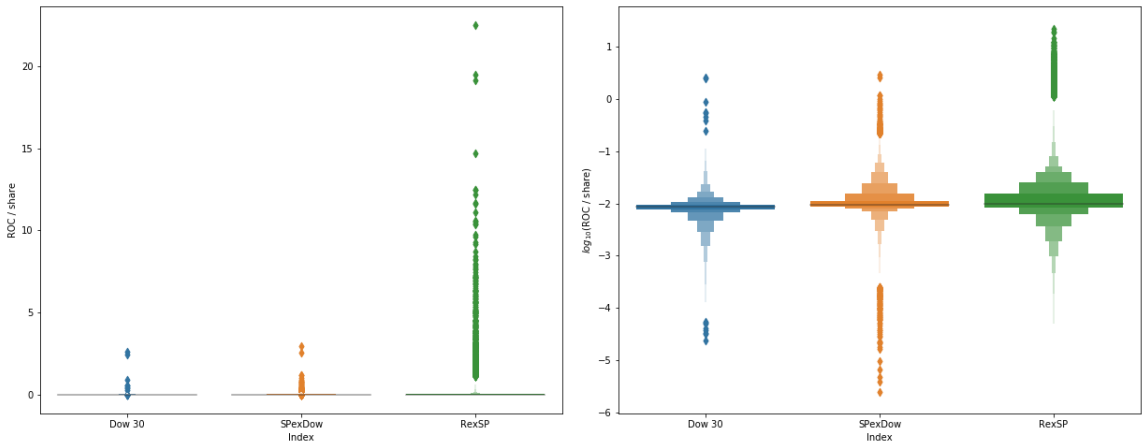


Figure A.14: Distributions of mean ROC per share per day ($ / day) over the members of each mutually exclusive market category. Linear (left) and log 10 (right) vertical axis scaling are used to provide additional perspective. On average, the members of the Dow experience the least ROC per share, followed by the SPexDow, followed by the RexSP.

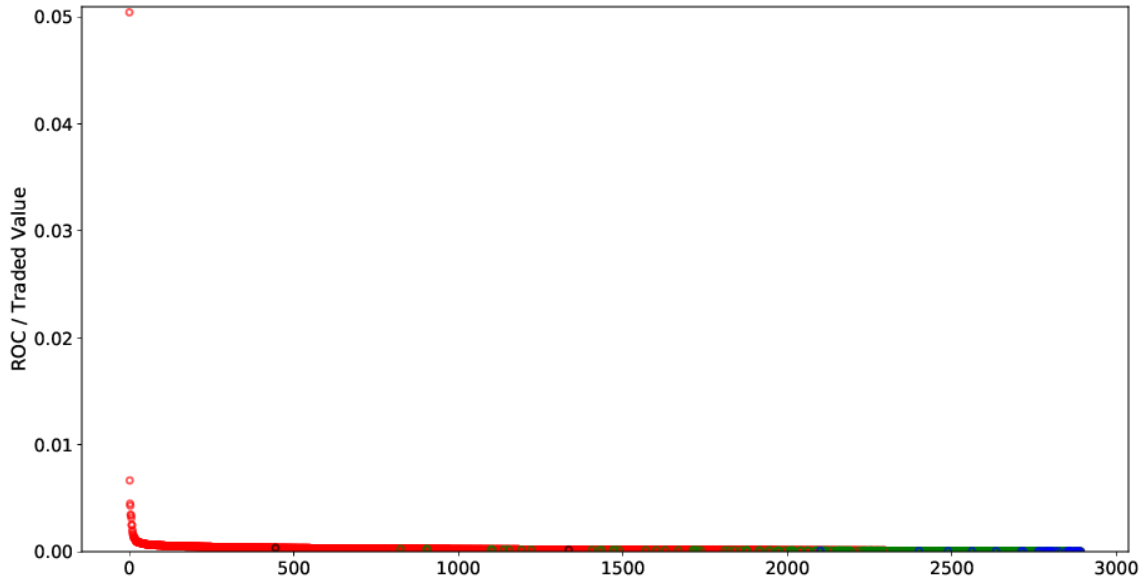


Figure A.15: Equities are plotted in rank-order of ROC per traded value; the 0-th equity has highest ROC per traded value. The first over-100 top equities are in the RexSP, which is unsurprising due to their combination of generally lower liquidity and lower share prices. Blue markers are associated with constituents of the Dow 30, green markers with constituents of the S&P 500 (excluding the Dow 30), red markers with constituents of the Russell 3000 (excluding the S&P 500), and black markers with ETFs.

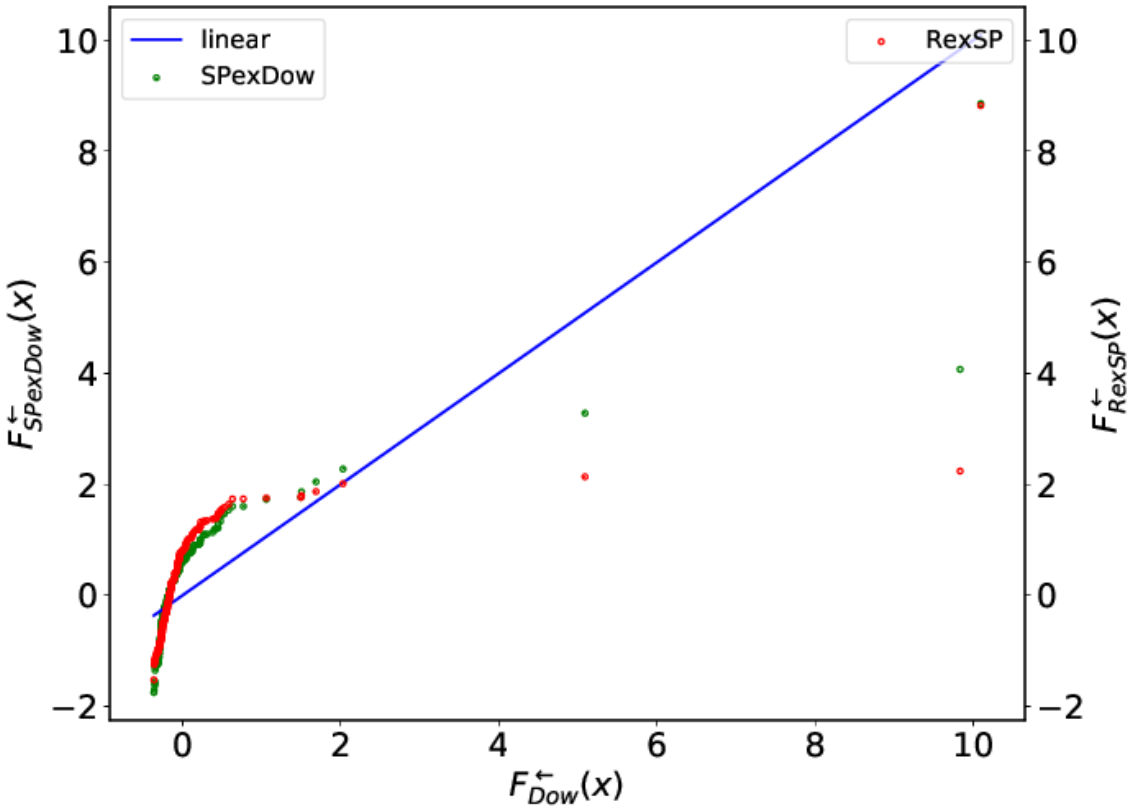


Figure A.16: Empirical quantile-quantile (QQ) plot for the normalized ROC per share processes. It is clear that the distribution of the SPexDow and RexSP processes are similar, and both are markedly different from the Dow process (blue line).

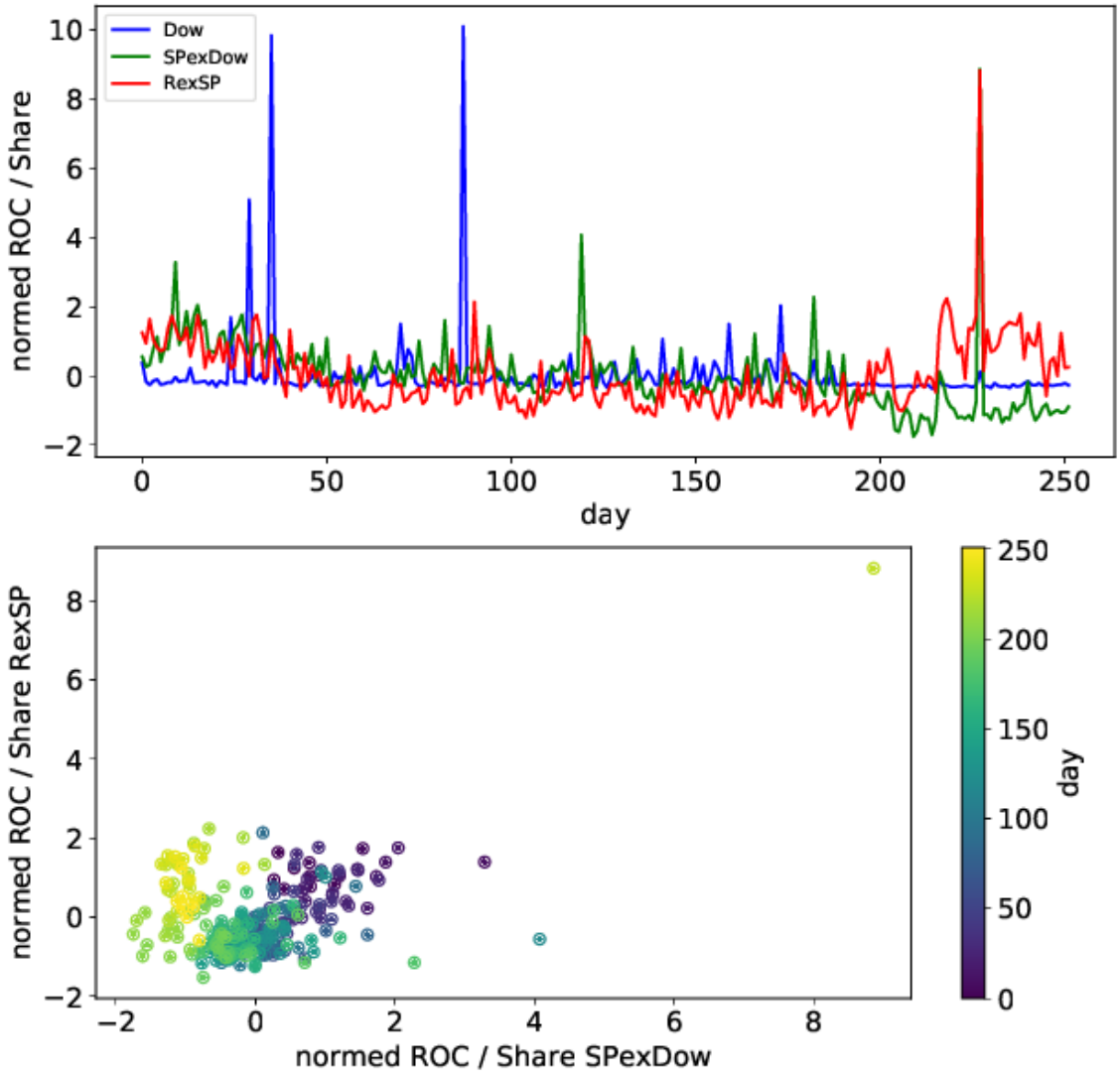


Figure A.17: Normalized ROC per share processes. There is one observation per day for a total of 252 observations in the process. These processes are anti-autocorrelated (Dow DFA exponent $\alpha = 0.434$, SPexDow DFA exponent $\alpha = 0.226$, RexSP DFA exponent $\alpha = 0.301$) and exhibit rare large values. The lower panel provides evidence for nonlinear cross-correlation between the SPexDow and RexSP ROC per share processes.

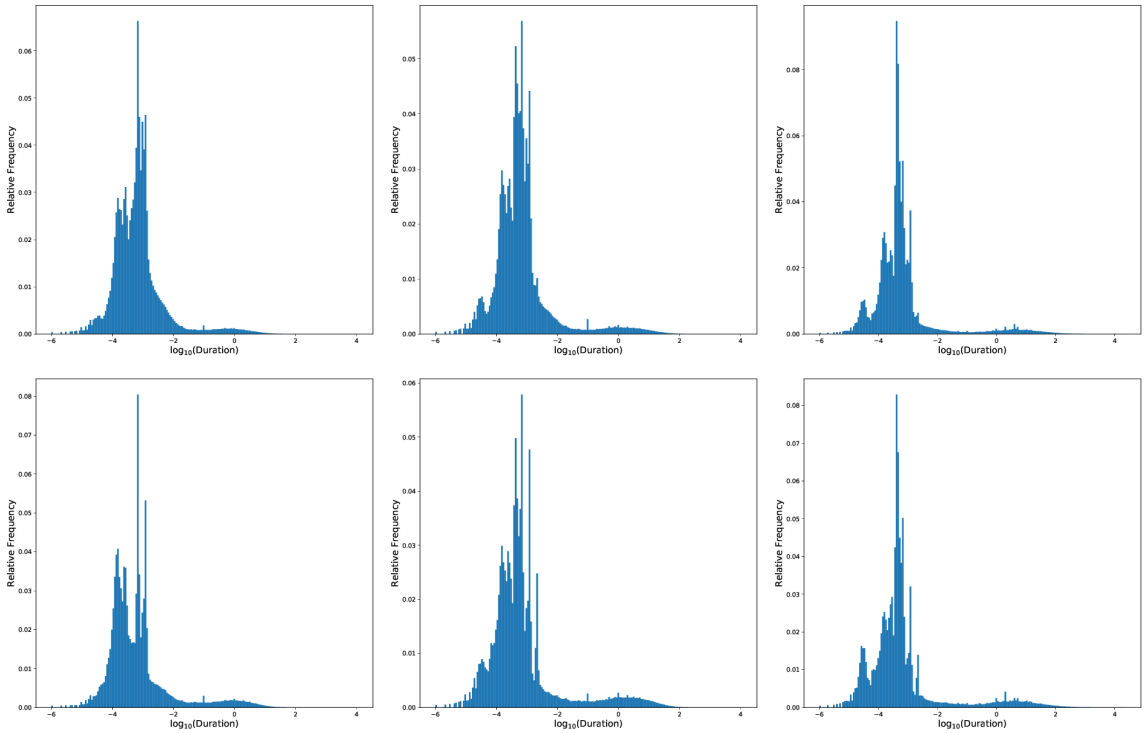


Figure A.18: Distributions of dislocation segment duration. Columns are associated with an index (left to right: Dow 30, S&P 500 excluding the Dow 30, Russell 3000 excluding the S&P 500) and rows are associated with conditioning strategies (top to bottom: no conditioning, magnitude greater than $0.01).

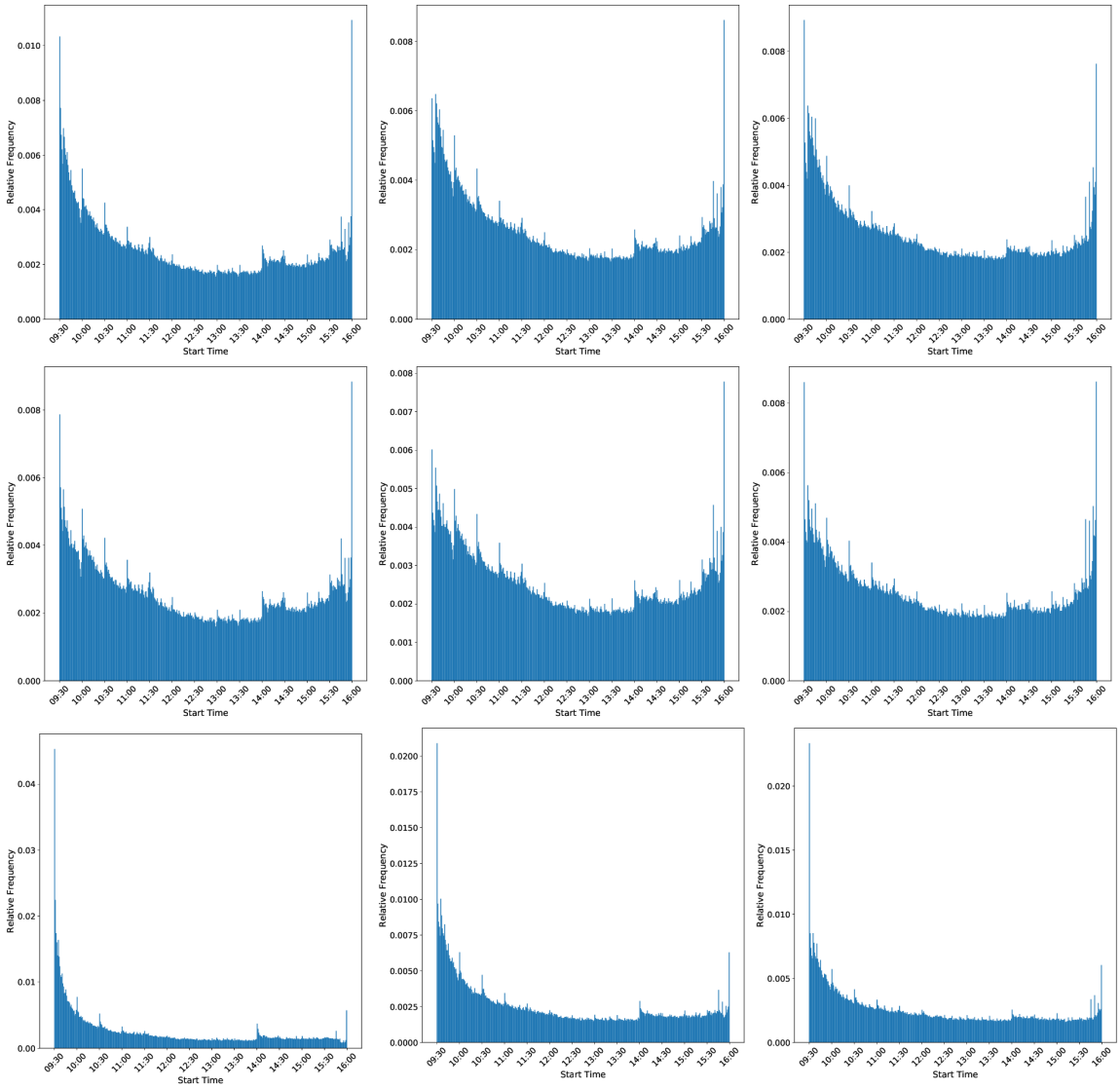


Figure A.19: Distributions of dislocation segment start time. Columns are associated with an index (left to right: Dow 30, S&P 500 excluding the Dow 30, Russell 3000 excluding the S&P 500) and rows are associated with conditioning strategies (top to bottom: no conditioning, duration greater than 545 μs, duration greater than 545 μs and magnitude greater than \$0.01).

A.5 NMS Statistics

Ordered pair	Lags
Dow → RexSP	2, 3, 4, 13, 14, 15, 20, 22, . . . , 37
Dow → SPexDow	
SPexDow → Dow	1, . . . , 10, 15,. . . ,24, 26, 30, . . . , 34
SPexDow → RexSP	1, 2, 3, 4
RexSP → Dow	1, 3, 35, 36
RexSP → SPexDow	

Table A.12: Granger causality results for pairwise combinations of mutually-exclusive market category under study. Statistical significance was assessed using four Granger causality tests (parameter F-test, sum of squared residuals F-test, likelihood-ratio test, χ^2-test). Each causal relationship was considered significant if each of the four tests resulted in a p-value less than $0.05/N_{lags}$. The maximum number of lags investigated was $N_{lags} = 40$.

Dep. Variable:	$\log_{10}$ ROC	**R-squared:**	0.908
Model:	OLS	**Adj. R-squared:**	0.908
Method:	Least Squares	**F-statistic:**	7179.
No. Observations:	2884	**Prob (F-statistic):**	0.00
Df Residuals:	2880	**Log-Likelihood:**	551.07
Df Model:	3	**AIC:**	-1094.
		BIC:	-1070.
Omnibus:	1630.431	**Durbin-Watson:**	2.007
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	23812.396
Skew:	2.375	**Prob(JB):**	0.00
Kurtosis:	16.252	**Cond. No.**	259.

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
Intercept	1.0052	0.091	11.050	0.000	0.827	1.183
l_MarketCap	0.1183	0.011	10.675	0.000	0.097	0.140
l_total_trades	-0.2203	0.043	-5.127	0.000	-0.304	-0.136
l_differing_trades	0.9023	0.040	22.286	0.000	0.823	0.982

Table A.13: Ordinary least squares regression predicting realized opportunity cost (ROC) using market capitalization, differing trades, and total trades.

Dep. Variable:	$\log_{10}$ ROC	**R-squared:**	0.925
Model:	OLS	**Adj. R-squared:**	0.925
Method:	Least Squares	**F-statistic:**	5970.
No. Observations:	2884	**Prob (F-statistic):**	0.00
Df Residuals:	2877	**Log-Likelihood:**	846.73
Df Model:	6	**AIC:**	-1679.
		BIC:	-1638.
Omnibus:	1952.210	**Durbin-Watson:**	1.988
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	50808.169
Skew:	2.831	**Prob(JB):**	0.00
Kurtosis:	22.768	**Cond. No.**	1.70e+04

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
Intercept	7.8666	0.802	9.811	0.000	6.295	9.438
l_MarketCap	-0.0738	0.149	-0.496	0.620	-0.365	0.218
l_total_trades	-4.1661	0.432	-9.638	0.000	-5.013	-3.319
l_differing_trades	3.0804	0.338	9.103	0.000	2.417	3.744
l_MarketCap ** 2	0.0067	0.008	0.837	0.402	-0.009	0.022
l_total_trades ** 2	0.3385	0.038	8.936	0.000	0.264	0.413
l_differing_trades ** 2	-0.2042	0.034	-6.002	0.000	-0.271	-0.138

Table A.14: Ordinary least squares regression predicting realized opportunity cost (ROC) using market capitalization, differing trades, and total trades. Quadratic terms are included.

Dep. Variable:	$\log_{10}$ ROC	**R-squared:**	0.600
Model:	OLS	**Adj. R-squared:**	0.600
Method:	Least Squares	**F-statistic:**	4280.
No. Observations:	2884	**Prob (F-statistic):**	0.00
Df Residuals:	2882	**Log-Likelihood:**	-1574.9
Df Model:	1	**AIC:**	3154.
		BIC:	3166.
Omnibus:	52.492	**Durbin-Watson:**	1.933
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	76.592
Skew:	0.199	**Prob(JB):**	2.34e-17
Kurtosis:	3.692	**Cond. No.**	126.

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
Intercept	-1.4415	0.108	-13.398	0.000	-1.652	-1.231
l_MarketCap	0.7368	0.011	65.422	0.000	0.715	0.759

Table A.15: Ordinary least squares regression predicting realized opportunity cost (ROC) using only market capitalization.

Dep. Variable:	$\log_{10}$ ROC	**R-squared:**	0.603
Model:	OLS	**Adj. R-squared:**	0.603
Method:	Least Squares	**F-statistic:**	2904.
No. Observations:	2884	**Prob (F-statistic):**	0.00
Df Residuals:	2881	**Log-Likelihood:**	-1564.7
Df Model:	2	**AIC:**	3135.
		BIC:	3153.
Omnibus:	67.584	**Durbin-Watson:**	1.935
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	100.782
Skew:	0.242	**Prob(JB):**	1.30e-22
Kurtosis:	3.777	**Cond. No.**	1.24e+04

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
Intercept	-6.2441	1.286	-4.857	0.000	-8.764	-3.724
l_MarketCap	1.7575	0.266	6.598	0.000	1.235	2.280
l_MarketCap ** 2	-0.0539	0.014	-3.927	0.000	-0.081	-0.027

Table A.16: Ordinary least squares regression predicting realized opportunity cost (ROC) using only market capitalization. Quadratic terms are included.